SHEAFPRUNE: SHEAF-GUIDED GRAPH PROPAGATION FOR VISUAL TOKEN PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models process the complete visual token sequence, increasing prefill cost even when many tokens are irrelevant to the instruction. Token-wise saliency captures limited inter-token dependence; graph-based selection instead depends on how relations govern importance propagation and ranking. We study support-constrained propagation with node-degree correction. Inspired by sheaf geometry, SheafPrune uses modality-specific maps and a sparse text–visual graph to transfer relevance, then propagates scores on a separate visual graph and corrects them for degree. It retains a hard top-k subset once before the frozen language model. The selector has about 4.5M parameters and is trained end-to-end through the frozen LLaVA-NeXT-8B backbone using only the language-modeling objective. Explicit retention-ratio conditioning lets one checkpoint serve every budget. Across four backbones, ten benchmarks and seven pruning ratios, with all baselines re-implemented under one protocol, SheafPrune achieves the best retention at 22 of 28 backbone–budget settings, with monotonically growing margins in the moderate-to-aggressive regime. On LLaVA-NeXT, retaining 10% of visual tokens yields a $5.15\times$ net prefill speedup and an 84.0% end-to-end FLOPs reduction. Together with component ablations, these results support graph-constrained importance propagation and degree correction as a design for visual token selection across the evaluated backbones and tokenization schemes, improving the trade-off between accuracy retention and inference cost.

1 INTRODUCTION

Multimodal large language models (MLLMs) encode an image into visual tokens and feed the visual sequence into a language model through a projector. Visual encoding covers the image independently of the current instruction, so redundant tokens pass through every decoder layer alongside tokens that carry essential evidence. Which tokens are redundant, however, depends on their relations and the question: a region dispensable for holistic recognition may contain an attribute or spatial cue; visually similar instances may jointly answer a counting question; a low-saliency region may carry the required text. A selector must therefore weigh tokens jointly under a finite retention budget, preserving useful evidence rather than simply keeping the largest individual scores.

Existing selectors draw on visual redundancy, language-model attention, or task-conditioned relevance. Graph methods, including G-Prune (Jiang et al., 2025), additionally relate visual tokens and propagate importance between them. For these methods, the initial scores do not determine the final ranking on their own. The graph specifies which tokens can exchange scores, while its connectivity affects how much each token receives. This motivates our central question: *under a fixed visual-token budget, how does graph connectivity turn initial importance into an effective ranking, and what roles do propagation constraints and degree correction play?* Answering this question requires resolving two key design issues. First, before relevance is defined, the states of visual tokens and the instruction, which come from different modalities, must be aligned so that they can be compared (Liang et al., 2022). Second, propagation must be constrained to specified relations, and the effect of neighborhood size must be corrected; otherwise, densely connected tokens can attain higher ranks through connectivity alone (Kipf & Welling, 2017; Tang et al., 2020).

Sheaf geometry addresses both issues through heterogeneous comparison and relation-constrained transfer: local states are compared after mapping them into shared edge spaces, and interactions

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

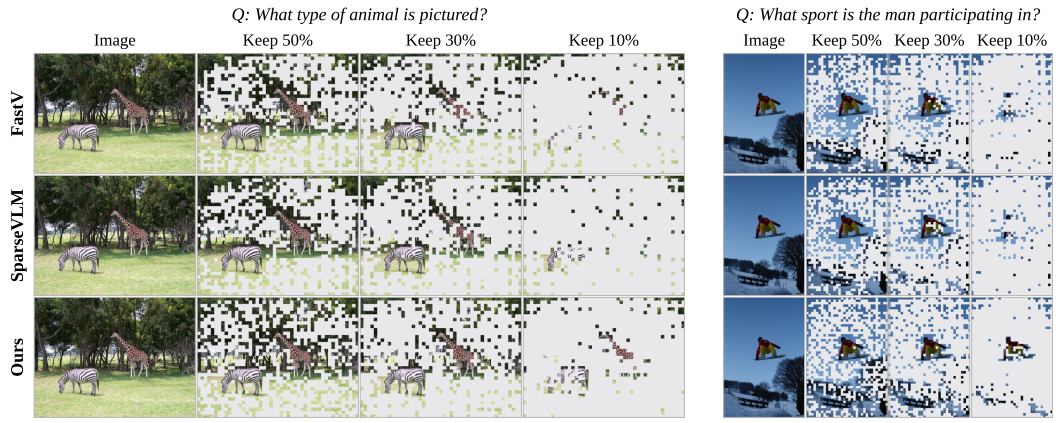


Figure 1: **Which regions survive pruning.** Two VQAv2 examples on LLaVA-NeXT-8B at retention ratios $\rho \in \{0.5, 0.3, 0.1\}$. FastV, SparseVLM and SheafPrune retain the same number of candidates; pruned positions are greyed out. The examples illustrate differences in retained regions, rather than establish the causal effect of question conditioning. Further samples appear in Appendix R.

follow specified relations (Hansen & Ghrist, 2019; Bodnar et al., 2022). Following this perspective, **SheafPrune** forms an initial score from cross-modal relevance and a visual prior, propagates it over a thresholded visual graph, corrects it for neighborhood degree so that connectivity alone does not inflate a token's rank, and retains a hard Top-k subset before the decoder. Modality-specific maps provide the comparison space; the visual stage uses an explicit graph transition rather than a learned sheaf Laplacian. Approximately 4.5M selector parameters are trained through a frozen backbone. Retention-ratio conditioning enables *one trained checkpoint to serve all evaluated budgets*, without training a separate selector for each budget.

Across four backbones and ten benchmarks, the reported per-point best-δ evaluation yields the highest mean retention at 22 of 28 backbone–budget settings. These are performance-envelope results; the fixed-configuration ablations answer a different question about the ranking operator. They show that uniform initialization preserves most of the full method's performance, whereas removing degree correction or changing the propagation operator substantially reduces retention. Selection also has a cost: on LLaVA-NeXT, retaining 10% of candidates yields a $5.15\times$ net prefill speedup, but at less aggressive budgets some simpler selectors are faster. Our contributions are:

- A sheaf-guided visual-token selector combining cross-modal comparison, support-constrained propagation and degree-corrected ranking, with one trained checkpoint covering multiple retention budgets.
- An analysis separating source-wise transition normalization from destination-wise degree correction, supported by fixed-configuration ablations across three budgets that identify the contribution of the combined ranking operator.
- An evaluation of accuracy retention and inference cost across four backbones, together with explicit hyperparameter-selection scope and coverage diagnostics that characterize limits of score-based selection.

2 Related Work

2.1 Visual Token Pruning for MLLMs

Visual compression methods differ in both their scoring signal and the operation applied to unselected tokens. VisionZip combines visual importance with feature similarity (Yang et al., 2025), DivPrune emphasizes subset diversity (Alvar et al., 2025), and ToMe merges similar tokens (Bolya et al., 2023). FastV uses shallow language-model attention (Chen et al., 2024); SparseVLM introduces textual signals (Zhang et al., 2025b); PyramidDrop removes tokens in stages (Xing et al.,

2024). G-Prune propagates visual importance over a similarity graph (Jiang et al., 2025), making it a direct comparison for graph-based ranking. Instruction- and interaction-aware methods include IVTP, LVPruning and ATP-LLaVA (Huang et al., 2024; Sun et al., 2025; Ye et al., 2025), while CD-Pruner and CoIn study relevance and representativeness at the subset level (Zhang et al., 2025a; Du et al., 2026). Class-attention methods include LLaVA-PruMerge and FasterVLM (Shang et al., 2025; Zhang et al., 2024). SheafPrune trains a small selector while keeping the backbone frozen. Our focus is how the graph operator transforms initial scores into a ranking; the experiments compare the evaluated implementations and do not establish superiority over untested selectors.

2.2 Sheaf Theory in Machine Learning

Cellular sheaves associate vector spaces with nodes and edges and use linear restriction maps to compare local states (Hansen & Ghrist, 2019). Neural Sheaf Diffusion learns these maps for graph representation learning (Bodnar et al., 2022); knowledge sheaves use typed comparisons for relational data (Gebhart et al., 2023). SheafPrune draws on this perspective to compare text and visual states in a shared space and to constrain information transfer to selected relations. Appendix A gives a conditional cellular-sheaf interpretation of the cross-modal edge energy. The subsequent visual ranking uses an ordinary graph transition and count-based correction. The contribution concerns their construction and evaluation for token selection, rather than a new general sheaf-diffusion operator.

3 Method

3.1 Problem Formulation

Let $X_T \in \mathbb{R}^{N_t \times d}$ and $X_V \in \mathbb{R}^{N_v \times d}$ denote text and visual representations at the selector interface. Masks $\boldsymbol{u} \in \{0,1\}^{N_t}$ and $\boldsymbol{v}, \boldsymbol{c} \in \{0,1\}^{N_v}$ identify valid text, valid visual positions, and prunable candidates, with $c_i \le v_i$. Structural positions, such as image-newline tokens, have $v_i = 1, c_i = 0$ and are retained outside the candidate budget. For $\mathcal{C} = \{i : c_i = 1\}$ and retention ratio ρ, the selector keeps exactly $k = \min\{|\mathcal{C}|, \max(1, \lceil \rho|\mathcal{C}| \rceil)\}$ candidates, plus mandatory structural positions. Appendix B defines the masks and empty-set handling. The notation d denotes the selector input width, not an assertion that every backbone has the same native hidden dimension.

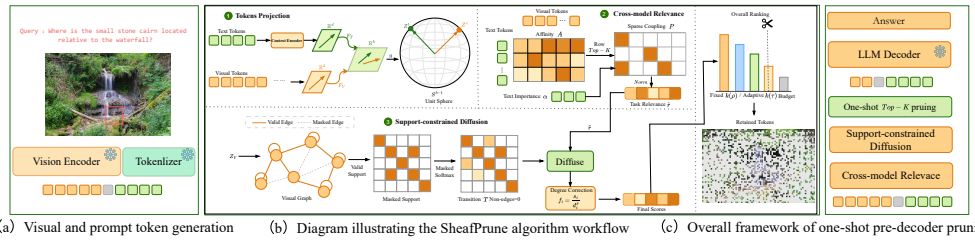
3.2 Sheaf-Guided Design

Two ideas from sheaf geometry guide the construction. First, a node carries a local representation, while an incident edge provides a space in which neighboring states can be compared through restriction maps. For text and visual tokens, type-specific maps therefore express compatibility without requiring the original feature spaces to share the same coordinates. Second, graph incidences specify which local comparisons and transfers are admitted. SheafPrune realizes these ideas through modality-specific comparison and sparse text–vision relevance transfer, followed by propagation on a separate visual graph. For fixed input-dependent normalizers, the cross-modal maps define linear restrictions and their squared disagreement is a sheaf edge energy (Appendix A). This correspondence motivates the comparison stage; the visual transition below transports scalar scores, and degree correction converts those scores into a ranking. Thus, *sheaf-guided* identifies the design rationale and its concrete realization without identifying the two stages with one sheaf Laplacian.

3.3 Budget-Conditioned Initialization

The textual mask admits the system prompt, completed dialogue turns and current question, but excludes target-answer positions, future turns, image placeholders, padding and control tokens. A lightweight bidirectional encoder produces $\widetilde{X}_T = \mathrm{Enc}_\phi(X_T; \boldsymbol{u}, \rho)$ outside the frozen language model, incorporating token order and the retention-ratio condition $g(\rho)$ (Appendix C). This keeps the selector's information boundary the same at training and inference.

After modality-specific LayerNorm, linear maps F_T, F_V project tokens into an h-dimensional comparison space. Write $\widehat{z} = z/(\|z\|_2 + \epsilon)$ for the stabilized normalized projection. Cross-modal affinity is $A_{ji} = \exp(-E_{ji}/\tau_E)$, with $E_{ji} = \|\widehat{z}^t_j - \widehat{z}^v_i\|_2^2 \in [0,4]$ and learned $\tau_E > 0$. For exact



(a) Visual and prompt token generation (b) Diagram illustrating the SheafPrune algorithm workflow (c) Overall framework of one-shot pre-decoder pruning

Figure 2: Overview of **SheafPrune**. (a) Frozen encoders supply visual and textual representations. (b) Modality-specific projection and sparse cross-modal transfer produce relevance; support-constrained visual propagation and degree correction transform the fused initial score into a ranking. (c) Hard Top-k selection physically removes unselected visual tokens before the frozen decoder. Matrices and scores are schematic.

unit vectors, this energy is $2 - 2\langle \widehat{z}_j^t, \widehat{z}_i^v \rangle$; Appendix D distinguishes that identity from stabilized normalization. The modality-specific maps allow text and visual states to be transformed differently before comparison, while normalization controls their scale; semantic usefulness is learned through the downstream objective.

A masked softmax gives textual importance α. Each valid text row retains its K strongest valid visual neighbors; degree normalization yields $P = D_T^{-1/2} \widetilde{A} D_V^{-1/2}$ and relevance $\widehat{r} = \mathrm{Norm}_{\boldsymbol{v}}(P^\top \alpha)$. Invalid positions are excluded before neighbor selection. The initial score fuses relevance with the normalized visual-feature norm $\overline{n}$:

$$s^{(0)} = (1 - \lambda)\overline{n} + \lambda \, \mathrm{std}_{\mathcal{C}}(\overline{n}) \, \mathrm{zscore}_{\mathcal{C}}(\widehat{r}), \qquad \lambda \in (0, 1). \tag{1}$$

The scale factor matches the relevance term to the visual prior. The prior supplies a visual-side signal, while the centered relevance term can promote or demote candidates relative to it. These are initial importance scores rather than probabilities; graph propagation acts on their relative values, and final selection uses their ordering. Question information enters through $\widehat{r}$, with the downstream graph determining how that information influences neighboring tokens.

3.4 Support-Constrained Propagation and Ranking

The visual graph is built from cosine similarities of valid visual features. Let ξ_{q_v} be their sample-wise $(1 - q_v)$ quantile over off-diagonal pairs. With explicit self-loops,

$$\mathcal{N}(i) = \{i\} \cup \{i' \neq i : v_{i'} = 1, \mathrm{sim}_{ii'} \geq \xi_{q_v}\}, \tag{2}$$

$$T_{ii'} = \frac{\mathbb{1}[i' \in \mathcal{N}(i)] \exp(\mathrm{sim}_{ii'})}{\sum_{l \in \mathcal{N}(i)} \exp(\mathrm{sim}_{il})}. \tag{3}$$

Non-edges have exactly zero transition weight; retained edges remain valid even when their cosine similarity is negative. The global threshold gives symmetric support while allowing different neighborhood sizes. Scores are propagated as row vectors, $s^{(\ell+1)} = s^{(\ell)} T$, for t steps, then corrected:

$$f_i = \frac{s_i^{(t)}}{d_i^{\delta}}, \qquad d_i = |\mathcal{N}(i)|. \tag{4}$$

Why normalization and correction differ. Row normalization fixes the total weight leaving each source; it does not equalize the total weight received by each destination. With uniform initialization $s_j^{(0)} = c$, one step gives

$$s_i^{(1)} = c \sum_j T_{ji}, \qquad f_i^{(1)} = \frac{c \sum_j T_{ji}}{d_i^{\delta}}. \tag{5}$$

Thus, graph connectivity can produce a nonuniform ranking even from uniform scores. Degree correction separately adjusts that ranking using neighborhood counts, not weighted row sums. This

explains how the operations differ; their usefulness is tested in Section 4.3. Appendix E analyzes longer-run degree bias and its limits for finite-step propagation. The two stages also use different measures of connectivity: T depends on exponential similarity weights, whereas d_i counts the retained neighbors. Dividing by d_i^δ therefore does not generally cancel the effects of propagation. Nodes with equal neighborhood counts can receive different propagated scores, while nodes with similar received mass can be reordered by correction. Neither operation explicitly enforces spatial coverage.

Algorithm 1 SheafPrune selection. Stages are italicized; every statistic, softmax, and ranking is evaluated only on its valid support.

Require: X_T, X_V; masks $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{c}$; retention ratio ρ
Ensure: mask m; retained visual indices $\mathcal{I}_{\text{keep}}$

Budget: $k \leftarrow \min\{|\mathcal{C}|, \max(1, \lceil \rho|\mathcal{C}| \rceil)\}$; $\widetilde{X}_T \leftarrow \text{Enc}_\phi(X_T; \boldsymbol{u}, \rho)$ ▷ Eq. (12), (15)
Relevance: $Z_T, Z_V \leftarrow$ modality maps; $A \leftarrow \exp(-E/\tau_E) \odot \boldsymbol{u}\boldsymbol{v}^\top$ ▷ Eq. (19)
$\widetilde{A} \leftarrow A \odot M_K$; $\widehat{r} \leftarrow \text{Norm}_{\boldsymbol{v}}(P^\top \alpha)$ ▷ Eq. (22), (23)
Propagation: $\mathcal{N}(i) \leftarrow \{i\} \cup \{i' : v_{i'} = 1, \text{sim}_{ii'} \geq \xi_{q_v}\}$; $T \leftarrow \text{Softmax}_{\mathcal{N}(i)}(\text{sim})$
$s^{(0)} \leftarrow (1-\lambda)\overline{n} + \lambda \, \text{std}_{\mathcal{C}}(\overline{n}) \, \text{zscore}_{\mathcal{C}}(\widehat{r})$ ▷ Eq. (28)
for $\ell = 0, \ldots, t-1$ **do** $s^{(\ell+1)} \leftarrow s^{(\ell)} T$ ▷ Eq. (29)
end for
$f_i \leftarrow s_i^{(t)} / d_i^\delta$, $d_i = |\mathcal{N}(i)|$ ▷ degree correction
Selection: $m \leftarrow \text{STTopK}(\text{zscore}_{\mathcal{C}}(f), k; \beta)$ ▷ Eq. (6)
$\mathcal{I}_{\text{keep}} \leftarrow$ selected candidates and mandatory positions in original order
return $m, \mathcal{I}_{\text{keep}}$; train: $p' \leftarrow \text{Gate}(p; m)$, infer: $X_V^{\text{keep}} \leftarrow \text{Gather}(X_V, \mathcal{I}_{\text{keep}})$

3.5 Training and Inference

Candidate scores are standardized and selected with a straight-through Top-k estimator (Bengio et al., 2013). The forward pass uses a hard subset, while the backward pass uses a smooth surrogate; hard selection itself is not differentiable. During training, the mask gates attention probabilities:

$$m = \text{STTopK}(\text{zscore}_{\mathcal{C}}(f), k; \beta), \qquad p'_{q\kappa} = \frac{\mu_\kappa p_{q\kappa}}{\sum_{\kappa'} \mu_{\kappa'} p_{q\kappa'} + \epsilon}. \tag{6}$$

Here μ_κ is the visual gate, or one for textual and mandatory structural positions. The selector is trained on LLaVA-NeXT-8B: the vision encoder, projector and language model remain frozen, and only the approximately 4.5M selector parameters are updated by target-answer language-modeling loss. Gradients pass through the frozen generation path and the multiplicative gate with renormalization. Training uses the LLaVA-NeXT instruction mixture(Li et al., 2024), weighted by the signal its sources provide for visual-token selection. After initial training at $\rho = 0.9$, retention-ratio sampling and conditioning through $g(\rho)$ train one checkpoint for all evaluated budgets (Appendix M). The selector sees the same admissible context during training and inference; target-answer positions are excluded before its text encoder. Appendix F and Appendix M specify the estimator, objective and training configuration.

At inference, the selector runs once before the decoder. Unselected visual tokens are physically removed, and retained tokens preserve their original position identifiers, including multidimensional coordinates where required by the backbone. Gating and physical removal share the same hard subset but remain distinct executions, so strict numerical equivalence is not assumed. All reported accuracy and efficiency results use the physical-removal path (Appendix G).

3.6 Space and Speed

One-shot removal shortens the sequence from the first decoder layer. Its net benefit depends on whether saved decoder computation exceeds selector overhead. Materializing the text attention, cross-modal affinity and visual similarity matrices requires auxiliary space $\mathcal{O}(N_t^2 + N_t N_v + N_v^2)$; sparse propagation alone does not eliminate the quadratic similarity workspace. All reported prefill latencies include selection cost. Training is an offline cost, while visual encoding and projection occur before pruning and are not avoided. For competing methods that also shorten the sequence once before the decoder, the remaining difference depends on selector kernels, memory access and

execution paths. The accuracy–latency trade-off is therefore an empirical system result under the stated measurement protocol. Appendix J gives the cost comparison.

4 EXPERIMENTS

4.1 SETUP

Backbones, benchmarks and metrics. We evaluate LLaVA-NeXT-8B (Li et al., 2024), Qwen2.5-VL-7B (Bai et al., 2025), InternVL3-8B (Zhu et al., 2025), and LLaVA-1.5-7B (Liu et al., 2024a). The ten benchmarks are ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), TextVQA (Singh et al., 2019), HR-Bench 4K/8K (Wang et al., 2025), GQA (Hudson & Manning, 2019), VQAv2 (Goyal et al., 2017), MMBench-EN (Liu et al., 2024b), POPE (Li et al., 2023) and MME (Fu et al., 2025). These benchmarks cover text-rich chart and document understanding (ChartQA, DocVQA, TextVQA), fine-grained perception in high-resolution images (HR-Bench 4K/8K), compositional reasoning and general visual question answering (GQA, VQAv2), holistic multimodal perception and cognition (MMBench-EN, MME), and object hallucination (POPE). All methods use the same benchmark splits and official metric definitions (Appendix K). RA% is the macro-average of ten pruned-to-unpruned score ratios; MME is normalized before aggregation. With pruning rate $p = 1 - \rho$, we evaluate $p \in \{0.1, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Prefill latency includes selection; LR% compares it with the unpruned reference.

Baselines. We use the baselines' open-source implementations and published method specifications within a common evaluation pipeline. Random, G-Prune, FastV, SparseVLM, PyramidDrop and ToMe are evaluated on all four backbones; VisionZip, PruMerge and FasterVLM are reported on LLaVA-1.5. Appendix L describes evaluation coverage and layer-averaged budget accounting. A shared pipeline standardizes data and metrics, while algorithm-specific pruning locations and execution costs remain different. Claims are restricted to the evaluated methods and settings.

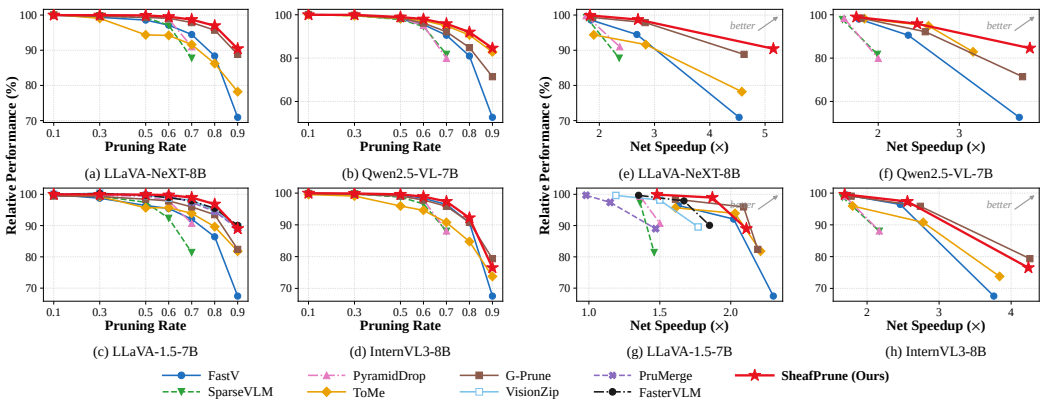


Figure 3: **Reported best-δ performance envelope and prefill trade-offs.** RA% averages ten benchmark retention ratios. The SheafPrune entries use per-point degree-exponent selection; they do not represent one fixed-δ deployment configuration. Unavailable staged-method points reflect the layer-averaged budget limits described in Appendix L.

Training and reporting protocol. Training has two stages: approximately 0.3 of an epoch on a strong-signal subset at $\rho = 0.9$, followed by approximately 2 epochs on the source-weighted 779K mixture. Stage B repeats four steps sampling ρ uniformly from $\{0.5, 0.3, 0.1\}$ followed by one additional step at $\rho = 0.9$. The ratio is supplied through $g(\rho)$; one checkpoint serves all seven evaluation budgets and is reused for the other backbones without retraining. The main comparison reports the best result over $\delta \in \{0, 0.5, 1, 2\}$ at each evaluation point: a performance envelope, not a fixed-δ deployment curve. The $22/28$ count and margin trends follow this protocol. Component ablations use a fixed default configuration, changing only the specified arm. Checkpoint reuse refers to trained weights; inference hyperparameters follow the stated reporting protocol. Appendix M gives the sampling, optimization and stage configurations.

4.2 Accuracy Retention and Inference Cost

Figure 3 summarizes the seven pruning rates; Table 1 reports per-benchmark scores at $p = 0.7$, and Appendix N gives all budgets. Under the reported best-δ protocol, SheafPrune leads all 16 backbone–budget points at $p = 0.5$–0.8. The reported margins over the strongest competitor increase across this range, from $+0.10$ to $+1.25$ on LLaVA-NeXT and $+0.22$ to $+1.40$ on InternVL3. These are comparisons of point estimates, not significance claims. Across all four backbones, the advantage is narrow at light pruning and widens consistently as the budget tightens.

Table 1: Comparison across four backbones (SheafPrune: reported best-δ protocol) at $p = 0.7$ ($\rho = 0.3$). acc: benchmark score (MME/28); lat: prefill latency including selection (ms); RA%: ten-benchmark macro-average retention; LR%: latency reduction, both against the unpruned backbone. Red/blue mark the best/second-best pruning method per column and backbone; Appendix K gives the profiling protocol and Table 9 the native scores and the remaining budgets.

		General VQA		MLLM benchmarks			Text-oriented VQA			High-resolution		Avg			
Model	Method	GQA	VQAv2	MME	POPE	MMB	ChartQA	DocVQA	TextVQA	HR-4K	HR-8K	acc	lat	RA%	LR%
LLaVA-NeXT-8B	No prune	65.0	82.1	38.4	83.6	77.0	69.7	72.6	66.7	51.2	46.1	65.2	149.4	–	–
	FastV	63.5	81.0	37.7	82.5	76.1	59.2	55.2	61.7	51.4	45.5	61.4	55.7	94.5	62.7
	SparseVLM	58.5	76.5	37.1	83.0	**76.8**	45.0	46.4	46.9	51.0	**46.1**	56.7	63.3	87.8	57.6
	PyramidDrop	62.2	80.8	37.8	**85.0**	76.1	45.4	49.3	55.5	**52.1**	45.6	59.0	63.0	91.0	57.8
	ToMe	64.5	80.8	37.8	82.4	76.3	52.1	48.7	60.5	50.0	42.5	59.6	**52.6**	91.6	**64.8**
	G-Prune	**64.9**	81.6	37.8	83.6	75.7	66.5	64.8	65.6	51.2	45.8	63.8	52.8	97.9	64.7
	SheafPrune (Ours)	64.8	**82.1**	**38.4**	83.6	76.2	**67.5**	**67.2**	**66.1**	51.2	46.1	**64.3**	55.3	**98.7**	63.0
Qwen2.5-VL-7B	No prune	58.7	83.1	44.4	84.3	83.3	87.8	94.2	84.3	71.4	63.9	75.5	113.6	–	–
	FastV	51.7	76.8	41.8	79.9	79.5	63.0	74.5	**82.3**	68.1	61.9	68.0	47.9	90.6	57.8
	SparseVLM	49.2	68.9	40.2	76.2	76.9	27.3	67.3	66.7	69.5	63.0	60.5	57.1	81.8	49.7
	PyramidDrop	48.1	71.5	36.4	74.9	71.3	41.0	67.6	69.1	63.2	54.6	59.8	56.8	79.9	50.0
	ToMe	**58.2**	**80.8**	**43.4**	83.3	**81.3**	73.1	84.7	79.5	68.1	60.9	71.3	**43.4**	94.9	**61.8**
	G-Prune	51.2	74.8	39.8	80.1	78.5	66.8	88.0	78.4	**72.8**	**64.5**	69.5	44.0	92.2	61.2
	SheafPrune (Ours)	57.0	80.0	41.3	**84.2**	81.2	**77.1**	**88.1**	79.1	71.4	63.5	**72.3**	45.8	**95.8**	59.7
LLaVA-1.5-7B	No prune	58.7	75.8	36.8	81.1	67.9	18.0	24.6	47.4	33.9	36.2	48.1	39.7	–	–
	FastV	53.9	71.8	35.1	73.8	66.5	15.8	18.3	42.2	33.6	36.1	44.7	19.7	92.1	50.5
	SparseVLM	49.8	64.6	34.7	72.9	65.6	13.7	10.6	21.8	35.2	34.1	40.3	27.2	81.4	31.5
	PyramidDrop	55.3	72.1	34.9	73.5	65.6	16.0	17.1	40.4	33.1	34.4	44.2	26.5	90.7	33.3
	ToMe	57.4	73.9	36.0	**79.7**	66.2	16.0	17.8	42.7	34.4	35.6	46.0	19.6	93.9	50.7
	G-Prune	55.1	72.9	36.2	75.5	65.2	17.1	22.6	46.5	33.1	36.1	46.0	**19.0**	96.0	**52.2**
	FasterVLM	55.9	74.3	36.6	76.4	66.8	17.6	22.7	45.5	**35.8**	**37.1**	46.9	23.8	97.9	40.1
	VisionZip	56.5	74.4	**36.6**	76.9	67.0	17.7	23.1	45.6	35.1	37.1	47.0	26.5	98.1	33.3
	LLaVA-PruMerge	56.7	74.4	36.6	77.7	66.7	17.0	22.5	45.8	34.8	36.8	46.9	34.5	97.4	13.0
	SheafPrune (Ours)	**57.7**	**74.9**	36.4	79.3	**67.3**	**17.9**	**24.2**	**47.1**	33.9	36.2	**47.5**	21.2	**99.0**	46.5
InternVL3-8B	No prune	61.3	79.3	44.8	89.6	87.0	86.5	92.5	82.3	70.0	61.9	75.5	128.7	–	–
	FastV	60.5	**79.1**	**44.3**	88.7	**86.2**	76.5	75.5	**81.4**	70.4	61.5	72.4	52.3	96.4	59.3
	SparseVLM	58.3	76.4	43.7	87.5	84.9	45.2	60.8	66.5	68.8	61.2	65.3	59.3	88.0	53.9
	PyramidDrop	58.7	76.2	41.4	86.8	81.3	60.6	62.4	68.8	64.5	57.5	65.8	59.3	88.1	53.9
	ToMe	60.5	77.8	42.8	88.6	85.8	63.6	62.1	72.4	67.4	57.4	67.8	**46.3**	90.8	**64.0**
	G-Prune	59.7	78.0	44.1	89.3	85.5	72.2	77.8	78.5	**71.0**	**63.2**	71.9	47.0	95.8	63.5
	SheafPrune (Ours)	**61.0**	78.0	43.3	**89.5**	86.0	**81.6**	**83.3**	79.1	70.0	61.6	**73.4**	50.3	**97.3**	60.9

Performance and exceptions. At $p = 0.7$ on LLaVA-NeXT, DocVQA scores are 67.23 for SheafPrune and 64.84 for the strongest evaluated baseline. At $p = 0.9$, SheafPrune has the highest reported RA on LLaVA-NeXT (90.42%) and Qwen2.5-VL (84.60%), but is overtaken by FasterVLM and VisionZip on LLaVA-1.5 and by G-Prune on InternVL3. It leads 22 of 28 settings overall under the envelope protocol. The four backbones span different tokenization schemes; the results support application across these evaluated interfaces.

Accuracy has a selection cost. At $p = 0.7$ on LLaVA-NeXT, SheafPrune retains 98.71% of performance, against 91.61% for ToMe and 97.86% for G-Prune, with net prefill speedups of 2.70×, 2.84× and 2.83×, respectively. Better retention therefore costs about 5% more latency than these two methods at this budget. At $p = 0.9$, the reported accuracy and speed optima coincide on LLaVA-NeXT (90.42% RA, 5.15× speedup) and Qwen2.5-VL (84.60%, 3.87×), but not on the other two backbones. The reported selector overhead is 2.6% of unpruned prefill on LLaVA-NeXT and 7.2% on the shorter LLaVA-1.5 sequence, where the $p = 0.9$ speedup is 2.11×. Across token counts and budgets, the selector costs at most a few percent of prefill while recovering several points of retention, and on the longer sequences it delivers the best accuracy and the best speedup at once.

Pruning location and inference cost. A fixed retained set shortens every decoder layer, whereas decoder-internal methods retain longer sequences in early layers. Under the budget accounting in Appendix L, nominal 90% pruning for FastV corresponds to 16.4% layer-averaged retention for $L = 28, K = 2$; the implemented staged schedules have a 25% floor. This distinction motivates reporting the layer-averaged budget alongside accuracy. Prefill speed and end-to-end FLOPs remain different quantities: the 84.0% FLOPs reduction on LLaVA-NeXT includes computation before pruning that cannot be removed and does not measure full-answer latency. At $p = 0.7$, reported decoding throughput changes from 66.5 to 68.3 tokens/s, while the three decoder-internal methods report 50.9–51.7 tokens/s. These are observations of the documented execution paths; Appendix K gives the timing workload and profiling corrections.

4.3 Ablation and Diagnosis

How the operator forms the ranking. Table 2 compares six configurations on LLaVA-NeXT using the same defaults. At $p = 0.9$, removing degree correction reduces RA from 89.39% to 72.80%; Dense normalization gives 72.68%, against 73.38% for random selection. In the documented dense variant, non-edge logits are zero before softmax and therefore contribute $\exp(0) = 1$, changing both effective propagation support and the denominator (Appendix P). It is an internal operator ablation, not a description of the G-Prune baseline. The similar losses do not establish identical failure mechanisms. Degree-only ranking reaches 81.38%: useful degree information alone does not recover the full ranking. On the text-oriented tasks, Dense normalization and no degree correction lose 36.87 and 36.67 retention points relative to the full method; degree-only ranking remains 17.88 points lower. The task-group columns therefore show a larger loss than the ten-task average, while the three budget columns show the pattern beyond a single pruning rate.

Table 2: **Fixed-configuration component ablation on LLaVA-NeXT-8B.** Defaults: $t = 3$, $q_v = 0.05$, $\delta = 1$, except as indicated. Norm.: the pair set used for normalization; Text: mean retention on ChartQA, DocVQA and TextVQA. Subscripts are differences from the full method at $p = 0.9$. RA averages ten benchmark retention ratios. These entries use fixed configurations, not the best-δ envelope; complete arms appear in Table 10.

			RA (%)			Text
Arm	Norm.	δ	$p = 0.5$	$p = 0.7$	$p = 0.9$	$p = 0.9$
Random selection	—	—	92.12	86.32	73.38 $_{-16.01}$	39.20 $_{-35.77}$
Degree only	—	1	95.77	92.14	81.38 $_{-8.01}$	57.09 $_{-17.88}$
Dense normalization	**dense**	1	91.74	85.78	72.68 $_{-16.71}$	38.10 $_{-36.87}$
No degree correction	support	**0**	91.88	85.97	72.80 $_{-16.59}$	38.30 $_{-36.67}$
Uniform initial score	support	1	98.50	96.72	88.19 $_{-1.20}$	73.32 $_{-1.65}$
SheafPrune	support	1	**99.43**	**97.97**	**89.39**	**74.97**

What initialization adds. Uniform initialization is 0.93, 1.25 and 1.20 RA points below the full method at the three budgets. This is consistent with Eq. (5): connectivity can create a ranking without an initially nonuniform signal. It does not show that training is unnecessary or that questions never affect selection. Rather, the tested operator retains most of the observed benefit, and these ablations do not establish learned initialization as the main gain source. Removing only relevance or only the visual prior performs worse than uniform initialization; Appendix P reports those interactions without assigning statistical significance to small differences.

What the operator computes. Figure 4 links the score-based analysis to two HR-Bench 8K examples. Regions with lower neighborhood degree receive higher corrected scores in these samples, with reported degree–score correlations of -0.854 and -0.931. This is consistent with the explicit division by d_i^δ, but propagated mass also enters the score: the figure is not evidence that low degree alone determines selection. The displayed retained regions illustrate the combined operator, without isolating the effect of question conditioning.

Graph construction. The existing k-NN comparison reports a 23–24-point decrease in the DocVQA score for tested $k \in \{16, 32, 64, 128\}$ (Appendix P.1). A sample-wise threshold permits variable neighborhood sizes while preserving symmetric support. Replacing it by k-NN changes

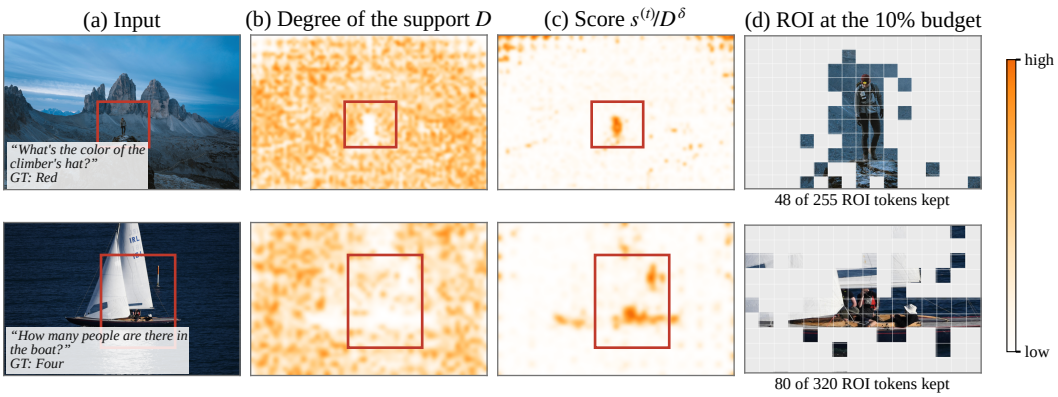


Figure 4: **Local score–degree relationships.** HR-Bench 8K examples at the 10% retention budget. Degree and corrected score are negatively associated in the displayed cases (reported correlations -0.854 and -0.931). Retained regions are shown for visual comparison. The existing visualization uses $t = 5$, $\delta = 1$; the default component table uses $t = 3$. Heatmaps are interpolated and scaled separately.

neighborhood structure and potentially symmetry and degree distribution together, so the comparison motivates examining graph construction but does not isolate one cause.

Where selection stops paying off. Mapping retained tokens back to the image grid reveals a limitation of score-based selection: neither propagation nor degree correction explicitly requires the selected subset to cover the image. At $p = 0.9$, GQA grid coverage is 0.417 for SheafPrune against 0.500 for random selection. Absolute counts also matter: 345 candidates leave about 35 tokens, whereas 4096 HR-Bench candidates leave about 410, with observed coverage 0.844. These examples distinguish a retention ratio from the amount and distribution of surviving evidence. They do not establish the cause of the LLaVA-1.5 or InternVL3 exceptions, because task content, tokenization and absolute counts differ simultaneously. Coverage-aware selection is a direction for testing, rather than an already validated remedy. Appendix Q provides the full coverage measurements.

Sensitivity within the tested setting. The existing Qwen2.5-VL sweep at $p = 0.9$ gives four-benchmark mean scores of 53.85–54.23 for $t \in \{1, 3, 5, 10\}$ and 53.06 for $t = 20$. Thus, additional propagation is not uniformly beneficial in this sweep. The separate absolute-similarity-threshold experiment has its largest displayed mean at $s = 0.5$; it should not be conflated with the default quantile parameter q_v. These measurements describe the tested settings and do not supply a noise threshold or a universal optimal depth (Appendix P.1).

5 Conclusion

SheafPrune uses sheaf-guided cross-modal comparison, support-constrained visual propagation and degree correction to rank visual tokens under a fixed budget. One trained checkpoint serves all evaluated retention ratios. The fixed-configuration ablations show that connectivity and the ranking operator preserve much of the full method's performance even from uniform initialization, while degree-only ranking is insufficient. Under the reported best-δ protocol, SheafPrune leads 22 of 28 backbone–budget settings; on LLaVA-NeXT, retaining 10% of candidates gives 90.42% RA, $5.15\times$ net prefill speedup and 84.0% end-to-end FLOPs reduction. These results characterize the measured accuracy–prefill trade-off across four backbones rather than a tuning-free deployment curve. Coverage observations further distinguish score quality from spatial coverage. Overall, SheafPrune advances graph-based visual token pruning through a sheaf-guided framework for budget-constrained selection, providing a methodological and empirical foundation for future multimodal compression methods that preserve useful visual evidence while reducing inference cost.

AI USE STATEMENT

In this work, we used generative AI tools, including Anthropic Claude Code with an Opus model and OpenAI Codex, to assist with code implementation, code review and testing, information retrieval, and critical feedback on the manuscript. The research ideas, algorithmic derivations, experimental design, and research plan were developed by the authors.

Claude Code assisted in implementing our designed functions and algorithmic components, including those involving sheaf theory and graph diffusion. AI tools also assisted with code review and testing to help identify potential bugs and gaps in test coverage. Codex assisted in finding related papers for comparison, locating datasets and model download sources, and troubleshooting network and system issues. We also used generative AI to provide critical feedback on the manuscript and help identify potential weaknesses in its reasoning and presentation.

We manually reviewed the AI-assisted code, tracing each function's logic from its inputs to its outputs and checking the algorithmic implementation against the mathematical formulations. All code was executed directly by the authors. Each paper identified through AI-assisted search was individually checked using search engines to verify its existence and guard against fabricated references.

We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our experiments use the publicly available LLaVA-NeXT-Data dataset(Li et al., 2024), which combines visual instruction-following data with document, chart, and diagram understanding data. According to the dataset documentation, the original dataset creators filtered samples with potential privacy concerns or harmful content from the 15K visual instruction subset collected through the LLaVA demo. We acknowledge that these upstream filtering measures do not eliminate all potential risks and that the dataset may retain biases, inaccuracies, or sensitive content inherited from its constituent sources. These limitations should be considered when interpreting the experimental results and assessing potential applications of the resulting models.

REPRODUCIBILITY STATEMENT

The formulation and algorithmic procedure of SheafPrune are presented in Section 3. Implementation settings and training configurations are described in Appendices I and M. The benchmark splits, evaluation metrics, generation settings, and efficiency measurement protocol are documented in Appendix K, while baseline implementations and token-budget accounting are described in Appendix L. The reporting protocol distinguishes per-point degree-exponent selection in the main comparisons from the fixed default configuration used for component ablations. Full reported benchmark scores and component ablation results are provided in Appendices N and P. Our code is included in the supplementary material. Model weights and checkpoints are omitted due to file-size limits. We will release the full source code and the trained selector checkpoints, including all weights, after the double-blind review.

REFERENCES

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9392–9401, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Alvar_DivPrune_Diversity-based_Visual_Token_Pruning_for_Large_Multimodal_Models_CVPR_2025_paper.html.

Stanislaw Antol et al. Vqa: Visual question answering. In *ICCV*, 2015.

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. URL https://arxiv.org/abs/2502.13923.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013. URL https://arxiv.org/abs/1308.3432.

Ali Furkan Biten et al. Scene text visual question answering. In *ICCV*, 2019.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Liò, and Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18527–18541, 2022. doi: 10.52202/068431-1346. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/75c45fca2aa416ada062b26cc4fb7641-Abstract-Conference.html.

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 (ICLR)*, 2023. URL https://arxiv.org/abs/2210.09461.

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 *Computer Vision – ECCV 2024*, volume 15139 of *Lecture Notes in Computer Science*, pp. 19–35. Springer, 2024. doi: 10.1007/978-3-031-73004-7_2. URL https://link.springer.com/chapter/10.1007/978-3-031-73004-7_2.

Chenxi Du, Yongheng Deng, Jiani Liu, Yujia Zhang, Xi Chen, and Ju Ren. CoIn: Coverage and informativeness-guided token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10492–10501, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Du_CoIn_Coverage_and_Informativeness-Guided_Token_Reduction_for_Efficient_Large_Multimodal_CVPR_2026_paper.html.

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 *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, 2025.

Thomas Gebhart, Jakob Hansen, and Paul Schrater. Knowledge sheaves: A sheaf-theoretic framework for knowledge graph embedding. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 9094–9116. PMLR, 2023. URL https://proceedings.mlr.press/v206/gebhart23a.html.

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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. URL https://arxiv.org/abs/1612.00837.

Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3(4):315–358, 2019. doi: 10.1007/s41468-019-00038-7. URL https://link.springer.com/article/10.1007/s41468-019-00038-7.

Kai Huang, Hao Zou, Ye Xi, BoChen Wang, Zhen Xie, and Liang Yu. IVTP: Instruction-guided visual token pruning for large vision-language models. In *Computer Vision – ECCV 2024*, volume 15075 of *Lecture Notes in Computer Science*, pp. 214–230. Springer, 2024. doi: 10.1007/978-3-031-72643-9_13. URL https://link.springer.com/chapter/10.1007/978-3-031-72643-9_13.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. URL https://arxiv.org/abs/1902.09506.

Yutao Jiang, Qiong Wu, Wenhao Lin, Wei Yu, and Yiyi Zhou. What kind of visual tokens do we need? training-free visual token pruning for multi-modal large language models from the perspective of graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(4):4075–4083, 2025. doi: 10.1609/aaai.v39i4.32427. URL https://ojs.aaai.org/index.php/AAAI/article/view/32427.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Bo Li, Kaichen Zhang, Hao Zhang, Dong Guo, Renrui Zhang, Feng Li, Yuanhan Zhang, Ziwei Liu, and Chunyuan Li. LLaVA-NeXT: Stronger LLMs Supercharge Multimodal Capabilities in the Wild. LLaVA project report, May 10, May 2024. URL https://llava-vl.github.io/blog/2024-05-10-llava-next-stronger-llms/.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL https://arxiv.org/abs/2305.10355.

Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y. Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17612–17625, 2022.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a. URL https://arxiv.org/abs/2310.03744.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision (ECCV)*, 2024b.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177/.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021. URL https://arxiv.org/abs/2007.00398.

Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *WACV*, 2020.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22857–22867, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Shang_LLaVA-PruMerge_Adaptive_Token_Reduction_for_Efficient_Large_Multimodal_Models_ICCV_2025_paper.html.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. URL https://arxiv.org/abs/1904.08920.

Yizheng Sun, Yanze Xin, Hao Li, Jingyuan Sun, Chenghua Lin, and Riza Batista-Navarro. LVPruning: An effective yet simple language-guided vision token pruning approach for multi-modal large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4299–4308. Association for Computational Linguistics, 2025.

doi: 10.18653/v1/2025.findings-naacl.242. URL https://aclanthology.org/2025.findings-naacl.242/.

Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. Investigating and mitigating degree-related biases in graph convolutional networks. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 1435–1444, 2020. doi: 10.1145/3340531.3411872.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(8):7907–7915, 2025. doi: 10.1609/aaai.v39i8.32852. URL https://ojs.aaai.org/index.php/AAAI/article/view/32852.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. PyramidDrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024. URL https://arxiv.org/abs/2410.17247v1.

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. 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, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ye_ATP-LLaVA_Adaptive_Token_Pruning_for_Large_Vision_Language_Models_CVPR_2025_paper.html.

Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [CLS] attention is all you need for training-free visual token pruning: Make VLM inference faster. *arXiv preprint arXiv:2412.01818*, 2024. URL https://arxiv.org/abs/2412.01818v1.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in MLLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-0855. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/2433fec2144ccf5fea1c9c5ebdbc3924-Abstract-Conference.html.

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.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. URL https://arxiv.org/abs/2504.10479v3.

A Sheaf-Theoretic Background and Method Positioning

The sheaf perspective guides heterogeneous comparison and relation-constrained transfer. A cellular sheaf on a graph assigns vector spaces to vertices and edges and linear maps from each incident vertex space to the edge space (Hansen & Ghrist, 2019). For an oriented edge $e = (u, v)$,

$$R_{v\to e} : \mathcal{F}(v) \to \mathcal{F}(e), \qquad (\delta_{\mathcal{F}} y)_e = R_{u\to e} y_u - R_{v\to e} y_v. \tag{7}$$

The squared edge disagreement and its aggregate are

$$\mathcal{E}_e(y) = \|R_{u\to e} y_u - R_{v\to e} y_v\|_2^2, \tag{8}$$

$$\mathcal{E}_{\mathcal{F}}(y) = \tfrac{1}{2} \sum_e \mathcal{E}_e(y) = \tfrac{1}{2} \|\delta_{\mathcal{F}} y\|_2^2 = \tfrac{1}{2} y^\top L_{\mathcal{F}} y, \qquad L_{\mathcal{F}} = \delta_{\mathcal{F}}^\top \delta_{\mathcal{F}}. \tag{9}$$

An associated linear diffusion has the form

$$y^{(\ell+1)} = (I - \eta L_{\mathcal{F}}) y^{(\ell)}. \tag{10}$$

This equation provides background, not the visual update used by SheafPrune.

Conditional correspondence for cross-modal comparison. Fix a sample, its text–vision edge set and its encoded, normalized local states $y_j = \mathrm{LN}_T(\widetilde{x}_j^t)$ and $y_i = \mathrm{LN}_V(x_i^v)$. Give each vertex the selector-interface space $\mathbb{R}^d$ and each edge $\mathbb{R}^h$. For that sample define the positive scalars $a_j = \|F_T y_j\|_2 + \epsilon$ and $a_i = \|F_V y_i\|_2 + \epsilon$, and set

$$R_{j\to e} = F_T / a_j, \qquad R_{i\to e} = F_V / a_i, \qquad E_{ji} = \|R_{j\to e} y_j - R_{i\to e} y_i\|_2^2. \tag{11}$$

Once the sample-dependent scalars are fixed, these are linear maps on a graph and specify a cellular sheaf. The evaluated edge energy is exactly the stabilized projected disagreement. The overall map from raw inputs to energy remains nonlinear because encoding, LayerNorm, normalization and edge selection depend on the input. This conditional correspondence does not turn the full selector into a fixed linear sheaf operator.

What this interpretation contributes. The correspondence makes the design choice explicit: text and visual states are compared after type-specific transformations into an edge space. With exact unit normalization, the same energy is also a cosine-based comparison of projected representations. This equivalence is acknowledged rather than treated as a separate source of expressivity. Unlike Neural Sheaf Diffusion (Bodnar et al., 2022), the visual stage here uses the independently constructed scalar transition in Eq. (3); it does not propagate with the cross-modal $L_{\mathcal{F}}$. Neighborhood-count correction is a further ranking operation. Sheaf guidance is therefore a design rationale with a precise cross-modal correspondence, while the empirical contribution is assessed through token-selection performance and operator ablations.

B Notation, Masks, and Fixed Budget

The masks of Section 3.1 are formed per sample: for sample b in a batch, with sequences $X_T^{(b)} \in \mathbb{R}^{N_t \times d}$ and $X_V^{(b)} \in \mathbb{R}^{N_v \times d}$, the textual validity mask $\boldsymbol{u} \in \{0,1\}^{N_t}$, the visual validity mask $\boldsymbol{v} \in \{0,1\}^{N_v}$ and the candidate mask $\boldsymbol{c} \in \{0,1\}^{N_v}$ with $c_i \le v_i$ handle variable-length text, AnyRes visual sequences and non-prunable structural positions such as `image_newline` ($v_i = 1$, $c_i = 0$). Given a target retention ratio $\rho \in (0, 1]$, the candidate set and retention budget of sample b are

$$\mathcal{C}^{(b)} = \left\{ i : c_i^{(b)} = 1 \right\}, \qquad k^{(b)} = \min\left\{ \left|\mathcal{C}^{(b)}\right|, \max\left(1, \left\lceil \rho \left|\mathcal{C}^{(b)}\right| \right\rceil\right) \right\}. \tag{12}$$

Because the number of candidates may differ across samples, $k^{(b)}$ is computed independently for each sample. Every softmax, Top-K operation, degree statistic, min–max normalization, z-score, and final ranking is evaluated only on its valid support.

C Leakage-Free, Budget-Conditioned Text Encoding

The selector receives only the system prompt, dialogue history completed before the target turn, and the current question. The current target answer, content following the target turn, expanded image-placeholder positions, padding, and configured control tokens are excluded. For each multi-turn

training sample, a target turn is first selected and the textual validity mask $\boldsymbol{u}$ is then constructed accordingly. The selector therefore has the same information boundary at training and inference and cannot exploit answer tokens to form a non-deployable pruning policy.

Because the frozen language-model input embeddings do not themselves contain explicit token order, we introduce a lightweight bidirectional encoder outside the language model. Its complete computation is

$$H_T^{(0)} = W_{\text{down}}\,\text{LN}_{\text{in}}(X_T) + P + \mathbf{1}\,g(\rho)^\top, \tag{13}$$

$$g(\rho) = \text{MLP}([\rho, \log\rho])\,, \tag{14}$$

$$H_T = \text{Enc}_{\text{bi}}\left(H_T^{(0)}; \boldsymbol{u}\right), \qquad \widetilde{X}_T = W_{\text{up}} H_T. \tag{15}$$

Here, $W_{\text{down}} \in \mathbb{R}^{d_e \times d}$ and $W_{\text{up}} \in \mathbb{R}^{d \times d_e}$ form a bottleneck projection, $P \in \mathbb{R}^{N_t \times d_e}$ is a learned absolute positional embedding, and $g(\rho) \in \mathbb{R}^{d_e}$ is the retention-ratio condition. The bidirectional encoder Enc_{bi} uses $\boldsymbol{u}$ as its padding mask and is independent of the causal attention layers in the frozen language model. We use a bottleneck dimension of $d_e = 256$, two bidirectional attention layers, four attention heads, and a maximum selector text length of 1024.

D Cross-Modal Compatibility and Text-to-Vision Propagation

Textual and visual tokens are projected into a shared h-dimensional space after modality-specific input normalization:

$$z_j^t = F_T\,\text{LN}_T(\widetilde{x}_j^t), \qquad \widehat{z}_j^t = \frac{z_j^t}{\|z_j^t\|_2 + \epsilon}, \tag{16}$$

$$z_i^v = F_V\,\text{LN}_V(x_i^v), \qquad \widehat{z}_i^v = \frac{z_i^v}{\|z_i^v\|_2 + \epsilon}. \tag{17}$$

The maps satisfy $F_T, F_V \in \mathbb{R}^{h \times d}$, with $h = 128$ by default. Post-projection ℓ_2 normalization bounds the cross-modal edge energy:

$$E_{ji} = \left\|\widehat{z}_j^t - \widehat{z}_i^v\right\|_2^2 = \|\widehat{z}_j^t\|_2^2 + \|\widehat{z}_i^v\|_2^2 - 2\left\langle \widehat{z}_j^t, \widehat{z}_i^v \right\rangle \in [0, 4]. \tag{18}$$

For exact unit vectors this reduces to $2 - 2\langle \widehat{z}_j^t, \widehat{z}_i^v \rangle$. With the stated additive ϵ, the squared norms are at most one and the full expansion above is the exact identity, including zero projections. The edge energy is converted into affinity by a negative-exponential kernel:

$$A_{ji} = u_j v_i \exp\left(-\frac{E_{ji}}{\tau_E}\right), \qquad \tau_E = \exp(\ell_\tau) > 0. \tag{19}$$

The temperature τ_E is initialized once from the median valid edge energy in the first minibatch and is subsequently optimized as an ordinary learnable parameter. Textual importance is defined by

$$a_j = w_{\text{imp}}^\top \widehat{z}_j^t + b_{\text{imp}}, \qquad \alpha_j = \frac{u_j \exp(a_j)}{\sum_{j'} u_{j'} \exp(a_{j'})}. \tag{20}$$

In the default path, α is used only as the textual propagation source and is not inserted into the exponential kernel in Eq. (19). This separation makes A_{ji} describe cross-modal geometric compatibility and α_j describe the contribution of textual token j to the current task.

For each valid textual token, only the K valid visual neighbors with the highest affinity are retained. The neighborhood size adapts to the visual sequence length:

$$K = \min\left\{N_v^{\text{valid}}, \max\left(K_{\min}, \left\lceil q_{\text{tv}} N_v^{\text{valid}} \right\rceil\right)\right\}. \tag{21}$$

The default values are $K_{\min} = 32$ and $q_{\text{tv}} = 0.05$. Invalid visual columns are masked before row-wise Top-K so that padding cannot occupy a valid neighbor slot. Let M denote the sparse support and $\widetilde{A} = A \odot M$. Bipartite propagation is then

$$D_T = \text{diag}(\widetilde{A}\mathbf{1}) + \epsilon I, \qquad D_V = \text{diag}(\widetilde{A}^\top \mathbf{1}) + \epsilon I, \tag{22}$$

$$P = D_T^{-1/2} \widetilde{A} D_V^{-1/2}, \qquad r = P^\top \alpha. \tag{23}$$

The relevance r is min–max normalized only over valid visual positions to obtain $\widehat{r}$. If $\max(r) - \min(r) < 10^{-6}$ over valid positions, we set $\widehat{r}_i = 0.5$ at all valid positions instead of amplifying numerical noise into an unstable ranking.

E VISUAL GRAPH, CONSTRAINED PROPAGATION, AND DEGREE CORRECTION

The visual graph is independent of the text–vision bipartite graph. For each pair of valid off-diagonal visual tokens, we first compute

$$\mathrm{sim}_{ii'} = \cos(x_i^v, x_{i'}^v). \tag{24}$$

Let ξ_{q_v} be the $(1-q_v)$ quantile of the valid off-diagonal similarities within a sample. The neighborhood of node i is

$$\mathcal{N}(i) = \{i\} \cup \{i' \neq i : v_{i'} = 1, \mathrm{sim}_{ii'} \geq \xi_{q_v}\}. \tag{25}$$

Self-loops are inserted explicitly and are excluded from quantile estimation, so q_v denotes the target retained proportion among valid off-diagonal relations. A sample-wise global threshold preserves a symmetric support for the undirected similarity graph. We use $q_v = 0.05$ by default and do not center the features before computing cosine similarity.

The transition matrix restricts propagation to the selected neighbors $\mathcal{N}(i)$:

$$T_{ii'} = \begin{cases} \dfrac{\exp(\mathrm{sim}_{ii'})}{\sum_{l\in\mathcal{N}(i)} \exp(\mathrm{sim}_{il})}, & i' \in \mathcal{N}(i), \\ 0, & i' \notin \mathcal{N}(i). \end{cases} \tag{26}$$

Edge existence is determined by the support in Eq. (25), not by whether the similarity value is positive. A retained edge with negative cosine similarity therefore still receives nonzero propagation weight, while every non-edge receives exactly zero weight.

The visual-norm prior is $n_i = \|x_i^v\|_2$. Let $\overline{n}$ denote its min–max normalization over valid visual positions. To avoid an empirical scale mismatch between the two branches, we standardize $\widehat{r}$ over the candidate set before early fusion:

$$\widetilde{r}_i = \frac{\widehat{r}_i - \mathrm{mean}_{\mathcal{C}}(\widehat{r})}{\mathrm{std}_{\mathcal{C}}(\widehat{r}) + \epsilon}, \tag{27}$$

$$s_i^{(0)} = (1-\lambda)\overline{n}_i + \lambda\, \mathrm{std}_{\mathcal{C}}(\overline{n})\, \widetilde{r}_i, \qquad \lambda = \sigma(\ell_\lambda). \tag{28}$$

Treating $s^{(\ell)}$ as a row vector, we perform propagation and degree correction as follows:

$$s^{(\ell+1)} = s^{(\ell)} T, \quad \ell = 0, \ldots, t-1, \qquad f_i = \frac{s_i^{(t)}}{d_i^\delta}, \qquad d_i = |\mathcal{N}(i)|. \tag{29}$$

Defaults are $t = 3$ and $\delta = 1$. Degree correction uses neighborhood counts, not weighted row sums or weight signs, and is distinct from transition-weight normalization.

FINITE-STEP REDISTRIBUTION AND LONG-RUN DEGREE BIAS

Let $W_{ij} = \mathbb{1}[j \in \mathcal{N}(i)] \exp(\mathrm{sim}_{ij})$, $w_i = \sum_j W_{ij}$ and $D_w = \mathrm{diag}(w_i)$. On valid visual nodes the symmetric similarity and global threshold give symmetric W, positive self-loops and $T = D_w^{-1} W$. For uniform scores, Eq. (5) follows by matrix multiplication: column sums, not row sums, determine the received score. Equal row sums therefore do not force equal rankings.

Long-run statement. For a connected component C, let $m_C = \sum_{j\in C} s_j^{(0)}$. Since each component is finite, connected and has positive self-loops, its random walk is irreducible and aperiodic. Its stationary distribution is $\pi_i = w_i / \sum_{j\in C} w_j$, and hence

$$\lim_{t\to\infty} s_i^{(t)} = m_C \frac{w_i}{\sum_{j\in C} w_j}, \qquad \lim_{t\to\infty} f_i = \frac{m_C}{\sum_{j\in C} w_j} \frac{w_i}{d_i^\delta}, \quad i \in C. \tag{30}$$

Proof. Symmetry gives detailed balance, $\pi_i T_{ij} = W_{ij} / \sum_{j\in C} w_j = \pi_j T_{ji}$. Irreducibility and aperiodicity imply $T_C^t \to \mathbf{1}\pi^\top$. Left-multiplication by the initial row vector gives the first limit; division by d_i^δ gives the second. This linear statement also holds for signed initial scores. If $m_C = 0$ the limiting component vanishes; if $m_C < 0$, its ranking direction is reversed.

For $\delta = 1$ and positive component mass, the limiting ranking within that component depends on w_i/d_i, the mean retained exponential similarity, rather than on neighborhood count alone. Count correction therefore changes degree dependence but need not eliminate all connectivity effects. Components may carry different total masses. The actual method uses finite $t = 3$ by default: these limits do not establish convergence within three steps, question independence, or an oversmoothing explanation for the measured accuracy losses. The finite-step identity and the ablations are the direct evidence used in the main text.

F Fixed-Budget Selection and End-to-End Training

The final score is first standardized over the candidate set:

$$\overline{f}_i = \frac{f_i - \mathrm{mean}_{\mathcal{C}}(f)}{\mathrm{std}_{\mathcal{C}}(f) + \epsilon}. \tag{31}$$

Let θ_k be the k-th largest standardized candidate score. The hard mask, smooth surrogate, and straight-through estimator are

$$m_i^{\mathrm{hard}} = \mathbb{1}[i \in \mathrm{TopK}_{\mathcal{C}}(\overline{f}; k)], \tag{32}$$

$$m_i^{\mathrm{soft}} = \sigma\left(\frac{\overline{f}_i - \mathrm{sg}(\theta_k)}{\beta}\right), \tag{33}$$

$$m_i = m_i^{\mathrm{soft}} + \mathrm{sg}\big(m_i^{\mathrm{hard}} - m_i^{\mathrm{soft}}\big), \tag{34}$$

where sg denotes stop-gradient. The forward pass uses the exact hard Top-k set, while the backward pass uses the sigmoid surrogate to pass gradients near the selection boundary. Exact ties are resolved using a fixed order of original token indices to ensure reproducibility. Every non-candidate structural token is assigned $m_i = 1$.

During training, visual gating is applied to the key dimension of every language-model attention layer. Let $p_{q\kappa}$ be the original attention probability from query position q to key position κ, and let μ_κ be the corresponding gate. The gated probability is

$$p'_{q\kappa} = \frac{\mu_\kappa p_{q\kappa}}{\sum_{\kappa'} \mu_{\kappa'} p_{q\kappa'} + \epsilon}. \tag{35}$$

Visual positions use their corresponding m_i, whereas textual and mandatory structural positions use $\mu_\kappa = 1$. Because at least the textual positions remain unpruned, the denominator of Eq. (35) remains positive.

Training uses only standard next-token cross-entropy at the target-answer positions:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} = -\frac{1}{|\mathcal{A}|} \sum_{q \in \mathcal{A}} \log p_\theta(y_q \mid y_{<q}, X_T, X_V; m), \tag{36}$$

where $\mathcal{A}$ is the set of current target-answer positions. The vision encoder, multimodal projector, and language model remain frozen, and only SheafPrune is updated. Gradients reach the selector through the frozen generation path and multiplicative gating. Here, end-to-end training means supervising SheafPrune directly with the final language-generation objective while keeping the LLaVA-NeXT backbone frozen.

G Inference-Time Physical Removal and Position Handling

Training-time gating provides gradients to the selector. At inference, unselected visual tokens are physically removed before entering the language model. Let $\mathcal{I}_{\mathrm{keep}}$ be the ordered index set containing all hard-selected candidates and mandatory structural positions. We compute

$$X_V^{\mathrm{keep}} = \mathrm{Gather}(X_V, \mathcal{I}_{\mathrm{keep}}), \qquad \mathrm{pos}^{\mathrm{keep}} = \mathrm{Gather}(\mathrm{pos}^{\mathrm{orig}}, \mathcal{I}_{\mathrm{keep}}). \tag{37}$$

The retained tokens preserve their original position indices instead of receiving newly generated contiguous `arange` indices, thereby maintaining the positional relations used by rotary embeddings. The attention mask, visual tokens, and position indices must be gathered with the same index set.

Training-time gating and inference-time physical removal use the same hard subset but remain different execution paths. In the current fp32 validation, the maximum logit difference is 5.147×10^{-3}, which remains above the strict 10^{-3} equivalence threshold. We therefore do not claim strict numerical equivalence between the two paths. All reported accuracy and efficiency measurements use the physical-removal path. Algorithm 2 instantiates Algorithm 1 with every appendix equation, default value, and support restriction made explicit.

Algorithm 2 Single-sample SheafPrune, with the defining equation of every step. Superscripts (b) are omitted; each statistic is evaluated only on its valid support.

Require: $X_T \in \mathbb{R}^{N_t \times d}$, $X_V \in \mathbb{R}^{N_v \times d}$; $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{c}$; ρ; defaults $h, K_{\min}, q_{\mathrm{tv}}, q_v, t, \delta, \beta$
Ensure: $m \in \{0,1\}^{N_v}$, ordered index set $\mathcal{I}_{\mathrm{keep}}$

Stage 1: budget and textual context

$\mathcal{C} \leftarrow \{i : c_i = 1\}, \quad k \leftarrow \min\{|\mathcal{C}|, \max(1, \lceil \rho|\mathcal{C}| \rceil)\}$ ▷ Eq. (12)

$\widetilde{X}_T \leftarrow W_{\mathrm{up}} \,\mathrm{Enc}_{\mathrm{bi}}\big(W_{\mathrm{down}} \,\mathrm{LN}_{\mathrm{in}}(X_T) + P + \mathbf{1} g(\rho)^\top; \boldsymbol{u}\big)$ ▷ Eq. (13)–(15)

Stage 2: cross-modal relevance

$\widehat{z}_j^t \leftarrow F_T \,\mathrm{LN}_T(\widetilde{x}_j^t)/(\|\cdot\|_2 + \epsilon), \quad \widehat{z}_i^v \leftarrow F_V \,\mathrm{LN}_V(x_i^v)/(\|\cdot\|_2 + \epsilon)$ ▷ Eq. (16), (17)

$E_{ji} \leftarrow \|\widehat{z}_j^t - \widehat{z}_i^v\|_2^2 \in [0,4], \quad A_{ji} \leftarrow u_j v_i \exp(-E_{ji}/\tau_E)$ ▷ Eq. (18), (19)

$\alpha_j \leftarrow u_j \exp(a_j) / \sum_{j'} u_{j'} \exp(a_{j'}), \quad a_j = w_{\mathrm{imp}}^\top \widehat{z}_j^t + b_{\mathrm{imp}}$ ▷ Eq. (20)

$K \leftarrow \min\{N_v^{\mathrm{valid}}, \max(K_{\min}, \lceil q_{\mathrm{tv}} N_v^{\mathrm{valid}} \rceil)\}, \quad \widetilde{A} \leftarrow A \odot M$ ▷ Eq. (21)

$P \leftarrow D_T^{-1/2} \widetilde{A} D_V^{-1/2}, \quad \widehat{r} \leftarrow \mathrm{MinMax}_{\boldsymbol{v}}(P^\top \alpha)$ ▷ Eq. (22), (23)

Stage 3: structure-constrained propagation

$\xi_{q_v} \leftarrow \mathrm{Quantile}_{1-q_v}(\mathrm{sim}_{i \neq i'}), \quad \mathcal{N}(i) \leftarrow \{i\} \cup \{i' : \mathrm{sim}_{ii'} \geq \xi_{q_v}\}$ ▷ Eq. (25)

$T_{ii'} \leftarrow \mathbb{1}[i' \in \mathcal{N}(i)] \exp(\mathrm{sim}_{ii'}) / \sum_{l \in \mathcal{N}(i)} \exp(\mathrm{sim}_{il})$ ▷ Eq. (26)

$s^{(0)} \leftarrow (1-\lambda)\overline{n} + \lambda \,\mathrm{std}_{\mathcal{C}}(\overline{n}) \,\mathrm{zscore}_{\mathcal{C}}(\widehat{r}), \quad \lambda = \sigma(\ell_\lambda)$ ▷ Eq. (27), (28)

for $\ell = 0, \ldots, t-1$ **do**

  $s^{(\ell+1)} \leftarrow s^{(\ell)} T$ ▷ Eq. (29)

end for

$f_i \leftarrow s_i^{(t)} / d_i^\delta, \quad \overline{f} \leftarrow \mathrm{zscore}_{\mathcal{C}}(f)$ ▷ Eq. (29), (31)

Stage 4: fixed-budget selection

$m_i^{\mathrm{hard}} \leftarrow \mathbb{1}[i \in \mathrm{TopK}_{\mathcal{C}}(\overline{f}; k)], \quad m_i^{\mathrm{soft}} \leftarrow \sigma\big((\overline{f}_i - \mathrm{sg}\,\theta_k)/\beta\big)$ ▷ Eq. (32), (33)

$m_i \leftarrow m_i^{\mathrm{soft}} + \mathrm{sg}(m_i^{\mathrm{hard}} - m_i^{\mathrm{soft}}), \quad m_i \leftarrow 1 \text{ if } c_i = 0, v_i = 1$ ▷ Eq. (34)

train: $p'_{q\kappa} \leftarrow \mu_\kappa p_{q\kappa} / (\sum_{\kappa'} \mu_{\kappa'} p_{q\kappa'} + \epsilon)$ ▷ Eq. (35)

infer: $\mathcal{I}_{\mathrm{keep}} \leftarrow \{i : m_i = 1\}, \quad (X_V, \mathrm{pos}) \leftarrow \mathrm{Gather}(\cdot, \mathcal{I}_{\mathrm{keep}})$ ▷ Eq. (37)

return $m, \mathcal{I}_{\mathrm{keep}}$

H Degenerate Cases and Numerical Stability

If a sample contains no visual tokens, the selector skips graph construction and follows the text-only path. If no prunable candidate exists, all valid structural tokens are retained. A sample with no valid textual token violates the current multimodal-input contract and is skipped and recorded. If $\rho = 1$ or $k = |\mathcal{C}|$, all candidate gates equal one, which also provides the unpruned path used for validation. If the variance of the candidate scores is nearly zero, ties are resolved deterministically by original token index so that different computational backends return the same subset.

Cross-modal energy, affinity, visual similarity, graph construction, and propagation are computed in fp32 to reduce underflow and ranking errors in large exponential kernels and softmax operations. After the selection mask is produced, it is converted to the hidden dtype used by the language model. Invalid positions are masked before normalization rather than multiplied by zero afterward; otherwise, invalid positions could first receive probability mass and alter the relative weights among valid positions.

I Implementation Configuration

Table 3 summarizes the default method configuration. Explicit retention-ratio conditioning allows one SheafPrune checkpoint to support multiple visual-token budgets. The bidirectional text encoder, modality-specific maps, textual-importance head, retention conditioner, and learnable scalar param-

eters introduce approximately 4.5M trainable parameters in total. The $d = 4096$ entry describes the training interface; it is not a specification of each evaluated backbone's native embedding width.

Table 3: Default SheafPrune configuration.

Component	Symbol	Default	Component	Symbol	Default
training-interface dimension	d	4096	visual-graph relation ratio	q_v	0.05
text bottleneck	d_e	256	propagation steps	t	3
text-encoder layers	L_e	2	degree exponent	δ	1
text-encoder heads	–	4	fusion mode	–	early
max selector text length	–	1024	visual cosine centering	–	false
comparison dimension	h	128	propagation normalization	–	masked
min bipartite neighborhood	$K_{\min}$	32	STE temperature	β	$2.0 \to 0.1$
bipartite ratio	q_{tv}	0.05	optimizer	–	AdamW
graph precision	–	fp32	initial learning rate	–	2×10^{-4}
trainable parameters	–	≈ 4.5M			

J Auxiliary Space and Speed Condition

When the textual self-attention matrix, text–vision affinity matrix, and vision–vision similarity matrix are directly materialized, the peak auxiliary space of the selector, excluding input and output representations and model parameters, is

$$M_{\text{SP}}^{\text{aux}} = \mathcal{O}\big(N_t^2 + N_t N_v + N_v^2\big)\,. \tag{38}$$

When $N_v \gg N_t$, the complete visual-similarity matrix is typically the dominant relational workspace. Sparsification lowers the practical storage constant during propagation, but it does not remove the N_v^2 term while the complete similarity matrix is explicitly constructed. Further reduction of peak space requires blockwise similarity computation or neighbor construction that does not materialize all pairwise relations.

Let $C_{\text{sel}}^{(a)}$ be the selection cost of method a, $n_\ell^{(a)}$ the sequence length processed by language-model layer ℓ, and $C_\ell(n)$ the cost of processing length n at that layer. Selecting once before the language model gives a constant length across layers, and SheafPrune is faster than method a whenever

$$n_\ell^{(\text{SP})} = N_t + N_v^{\text{keep}}, \qquad N_v^{\text{keep}} = k + N_{\text{str}}, \qquad \forall \ell \in \{1, \dots, L\}, \tag{39}$$

$$C_{\text{sel}}^{(\text{SP})} - C_{\text{sel}}^{(a)} < \sum_{\ell=1}^{L} \Big[C_\ell\Big(n_\ell^{(a)}\Big) - C_\ell\big(N_t + N_v^{\text{keep}}\big) \Big]\,. \tag{40}$$

Latency therefore depends jointly on selector overhead and the accumulated language-model computation saved by earlier sequence shortening. SheafPrune determines the subset once before the decoder, allowing the compressed sequence to be used from the first layer and avoiding layer-wise rescoring, token recycling, or online calibration. For methods that also perform one-shot physical removal before the decoder, the downstream language-model cost is approximately matched under the same budget, and the speed difference is governed primarily by the actual kernels, memory access, dynamic branches, and synchronization in the selector. We therefore report the speed advantage over the compared training-free methods as an empirical system result under a unified hardware and implementation protocol, rather than as a universal consequence of asymptotic complexity. The selector's own training is a one-off offline cost that does not enter the inference path, and the reported latencies include its forward cost.

The measured total includes the selector's forward cost. A full stage-level breakdown would additionally separate text encoding, cross-modal scoring, graph construction, propagation and Top-k; no such breakdown is inferred from the aggregate values. The vision encoder and multimodal projector execute before pruning, so their computation is not reduced by token removal.

K Evaluation Protocol

Benchmark sources. We use ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), TextVQA (Singh et al., 2019), HR-Bench 4K/8K (Wang et al., 2025), GQA (Hudson & Manning,

2019), VQAv2 (Goyal et al., 2017), MMBench-EN (Liu et al., 2024b), POPE (Li et al., 2023), and MME (Fu et al., 2025). The split and metric used for each benchmark are specified below.

Benchmarks. Table 4 lists the ten benchmarks used in our evaluation. Every method, backbone, and budget configuration is evaluated on the identical splits reported there, so that all comparisons in this paper are within-set comparisons. All metrics are re-implemented following the official evaluation protocol of each benchmark, and the same implementation is used for every method, so that reported scores are directly comparable.

Table 4: Evaluation benchmarks. All methods share the same sets and the same metric implementation.

Group	Benchmark	#Samples	Split	Metric
OCR / document / chart	ChartQA	2,500	test	relaxed accuracy
	DocVQA	5,349	validation	ANLS
	TextVQA	5,000	validation	VQA accuracy
	HR-Bench 4K	800	full	option accuracy
	HR-Bench 8K	800	full	option accuracy
General VQA / perception	GQA	12,578	testdev-balanced	exact-match accuracy
	VQAv2	214,354	validation	VQA accuracy
	MMBench-EN	4,377	dev	option accuracy
	POPE	9,000	adv./pop./rand.	F1
	MME	2,374	test	official MME score

Metrics. Each benchmark is scored with its official metric definition, summarized below and listed per benchmark in Table 4. Throughout, p denotes the model prediction after extraction and g the reference answer.

- **Relaxed accuracy** (ChartQA (Masry et al., 2022)). Introduced for numeric chart question answering (Methani et al., 2020): a numeric prediction is counted as correct if its relative error with respect to the reference is at most 5%, while non-numeric predictions require exact string match. The reported score is the fraction of correct predictions.
- **Average Normalized Levenshtein Similarity (ANLS)** (DocVQA (Mathew et al., 2021)). Proposed for text-based visual question answering (Biten et al., 2019) to tolerate optical-character-recognition errors while penalizing semantically wrong answers. With d_L the Levenshtein edit distance (Levenshtein, 1966) and $\mathrm{NL}(p, g) = d_L(p, g)/\max(|p|, |g|)$, the per-question score is $1 - \mathrm{NL}(p, g)$ if $\mathrm{NL}(p, g) < \tau$ and 0 otherwise, with $\tau = 0.5$; the reported score averages over all questions. Multiple reference answers are handled by taking the maximum over references.
- **VQA accuracy** (VQAv2 (Goyal et al., 2017), TextVQA (Singh et al., 2019)). The soft agreement metric of Antol et al. (2015): $\mathrm{Acc}(p) = \min(n(p)/3,\, 1)$, where $n(p)$ is the number of the ten human annotators whose answer equals p after the official normalization of articles, punctuation, and number words. The final value averages this quantity over all ten nine-annotator subsets.
- **Exact-match accuracy** (GQA (Hudson & Manning, 2019)). Answers are drawn from a closed vocabulary, so a prediction is correct only if it matches the reference exactly after the official normalization.
- **Option accuracy** (MMBench-EN (Liu et al., 2024b), HR-Bench 4K and HR-Bench 8K (Wang et al., 2025)). Both benchmarks are multiple choice and are scored under the cyclic evaluation protocol of (Liu et al., 2024b): each question is presented once per cyclic permutation of its options, and the question is counted as correct only if the model selects the reference option under every permutation. This removes the option-position bias that a single presentation would leave unmeasured.
- **F1** (POPE (Li et al., 2023)). Object-hallucination probing is cast as binary yes/no classification with `yes` as the positive class; the reported value is the harmonic mean of precision and recall, computed separately for the random, popular, and adversarial sampling settings.
- **Official MME score** (MME (Fu et al., 2025)). Each of the fourteen subtasks contributes $\mathrm{acc} + \mathrm{acc}^{+}$, where acc is per-question accuracy and acc^{+} is the fraction of images for

which both associated questions are answered correctly, each expressed as a percentage. The perception score sums the ten perception subtasks and the cognition score sums the four cognition subtasks.

Because the evaluated models emit free-form dialogue rather than bare answers, a shared extraction stage removes conversational prefixes and other non-answer text before scoring, and every evaluation record stores the raw generation alongside the extracted answer. This stage is necessary because the metrics above operate at the string level. For $g = \texttt{1963}$ under ANLS, the raw generation `The answer is 1963` yields $d_L = 14$ and $\max(|p|, |g|) = 18$, hence $\mathrm{NL} = 14/18 > \tau$ and a score of 0, whereas the extracted answer `1963` yields $\mathrm{NL} = 0$ and a score of 1. All metrics are re-implemented following the official protocol of each benchmark, and the identical extraction stage and metric implementation are applied to every compared method, so the benchmark metrics themselves are left unchanged.

Generation and image processing. Decoding is greedy for reproducibility, the generation length limit is set per benchmark, and the prompt suffix follows the convention of the training data of each backbone (a single-word/phrase instruction for open-ended VQA, an option-letter instruction for multiple choice) so that no arm loses points to instruction-format mismatch. Two backbone-side settings are fixed after measurement rather than by default: Qwen2.5-VL caps the visual sequence at 4096 tokens, and InternVL3 enables dynamic tiling with the official maximum of 12 patches. The latter is not the library default; leaving it off silently reduces a 4992×7680 image from 1792 to 256 visual tokens, which shows up only as a slightly lower score.

Determinism and sharding. Evaluations are split into eight shards with deterministic construction and index-based Top-k tie handling. All shards must load the same trained selector parameters, including the learned energy temperature τ_E; construction seeds are not a substitute for checkpoint loading. First-minibatch temperature initialization belongs to training initialization and must not silently replace the learned value during evaluation.

Correctness gates. The physical-removal path is checked at full retention against the unpruned generation path, with the retained count and position identifiers inspected. In particular, multidimensional positional information must be preserved for backbones that use it. G-Prune index selection is compared with its official implementation on the checked samples. These checks establish local implementation consistency rather than universal numerical equivalence. Run summaries surface per-sample failures so that a failed shard cannot silently reduce the set of evaluated samples.

Efficiency measurement. Latency is measured separately from accuracy. The efficiency record uses 15 samples each from GQA, ChartQA, DocVQA and HR-Bench 4K (60 total), with three warm-up runs and ten CUDA-event repetitions per sample. It reports prefill means, median selector overhead and greedy decoding over a forced 32-token continuation. The reported comparison covers retention ratios 0.5, 0.3 and 0.1, where the method's budget convention permits a run. Structural tokens remain outside the candidate budget; for example, $p = 0.9$ can retain 11.6% of all visual positions rather than exactly 10%.

Method-specific execution paths are important. One-shot selectors are timed after physical deletion or merging. For decoder-internal methods, the existing measurement uses one-token generation and a backbone-specific correction for the different language-model output-head computation. The recorded corrections are $+6.53$, $+4.70$, $+7.61$ and -0.36 ms for LLaVA-NeXT, Qwen2.5-VL, InternVL3 and LLaVA-1.5, respectively; these are measurement-protocol corrections, not algorithmic speedups. Thus, the table compares the documented profiling paths, and raw first-token latency should not be confused with the corrected prefill quantity. Per-sample speedup ratios are aggregated, rather than calculated by dividing independently aggregated medians.

FLOPs are reported for the language model alone and for the end-to-end computation including visual processing. Decoder-internal pruning uses layer-specific sequence lengths in the FLOPs calculation. Neither FLOPs reduction nor prefill speedup is a measurement of complete-answer wall-clock latency. Selector overhead, sequence length and execution path all affect the observed timing.

L Baseline Implementations and Budget Accounting

Why every baseline is re-run. Every baseline is reproduced from the authors' open-source implementation and published method specification inside our common evaluation pipeline, rather than transcribed from the scores reported in the original papers. All methods are thus measured under the same splits, the same metric implementation, and the same answer-extraction stage. Re-running is necessary because published scores depend on implementation details that the papers do not fully determine, including the repository revision, the decoding configuration, the numerical precision, and the answer post-processing; scores obtained along different execution paths are therefore not directly comparable even when the benchmark and the split coincide. Our baseline numbers may consequently deviate slightly from the published ones, but the differences we report between methods are attributable to the methods themselves rather than to their evaluation harnesses.

Budget accounting. One-shot methods reduce the visual sequence once before the language model by deletion or merging, so every layer processes the retained fraction ρ. FastV, PyramidDrop and SparseVLM prune inside the decoder, and their early layers still process the full sequence. Aligning budgets by the final token count would therefore credit them with compute they do not save, and the discrepancy grows with the pruning rate. We align by layer-averaged retention:

$$\text{FastV:}\quad \bar{\rho} = \frac{K + (L-K)r}{L}, \qquad \text{PyramidDrop:}\quad \bar{\rho} = \frac{1}{S}\sum_{j=0}^{S-1} \lambda^j. \tag{41}$$

We solve for the method-specific parameter r or λ so that $\bar{\rho}$ equals the target budget. For the latter, we use bisection because the sum is monotone on $\lambda \in [0, 1]$. SparseVLM is handled the same way, with recycled tokens counted as retained because they occupy sequence positions and are attended to. Under this convention, nominal 90% pruning for FastV ($L = 28$, $K = 2$) corresponds to 16.4% layer-averaged retention. Runs under the nominal convention are archived rather than deleted, so the question of whether the accounting changes any conclusion can be answered with data.

Compared methods and evaluation coverage. Table 5 summarizes the roles of the evaluated baselines. Table 6 distinguishes unavailable evaluation coverage from the mathematical budget floor of staged pruning. Unreported entries do not imply that every missing backbone–method combination is architecturally impossible.

Table 5: Evaluated baseline roles. Implementations follow the corresponding open-source releases and method specifications; measurement and budget conventions are stated separately.

Method	Role in the comparison
G-Prune	Visual graph propagation and ranking.
FastV	Decoder-attention-based token selection.
PyramidDrop	Staged removal within the decoder.
SparseVLM	Text-informed visual token sparsification.
ToMe	Token merging rather than pure deletion.
VisionZip	Visual importance and contextual compression.
FasterVLM	Class-attention-based token deletion.
LLaVA-PruMerge	Selection combined with merging.

Table 6: Reported evaluation coverage. Budget ranges use pruning rate p.

Method	Backbones	Budgets	Scope
FastV, ToMe, G-Prune	All four	0.1–0.9	Evaluated across the reported grid.
PyramidDrop, SparseVLM	All four	0.1–0.7	Under the implemented staged schedule, layer-averaged retention has a floor of $1/S = 0.25$.
FasterVLM, VisionZip, PruMerge	LLaVA-1.5	0.1–0.9	Reported for this backbone; the present comparison does not evaluate their other backbone integrations.

M TRAINING CONFIGURATION

What was trained. The selector is trained end-to-end through LLaVA-NeXT-8B. The vision encoder, projector and language model remain frozen; only the selector's approximately 4.5M parameters are updated. Training uses only next-token cross-entropy at target-answer positions, as specified in Appendix F. Gradients from the task loss reach the selector through the frozen generation path and the multiplicative gate in Eq. (6).

Because the mask is computed before the language-model forward pass, the selector's textual context excludes the target answer. For single-turn samples, supervised positions are removed. For multi-turn samples, one target turn is sampled per epoch; loss is evaluated on that turn, and the context contains the system prompt, completed turns and current question. The information boundary is defined by token positions rather than answer-string matching.

Two-stage retention-ratio training. Table 7 summarizes the training schedule. Stage A uses the strong-signal subset for approximately 0.3 of an epoch at fixed $\rho = 0.9$ and STE temperature $\beta = 2.0$. This stage provides initial training under a loose token budget. Stage B uses the source-weighted 779K mixture for approximately 2 epochs. Each cycle consists of four steps in which ρ is sampled independently and uniformly from $\{0.5, 0.3, 0.1\}$, followed by one additional step at $\rho = 0.9$. Thus, the loose-budget step occurs once per five-step cycle and does not replace one of the four randomly sampled steps. The STE temperature follows cosine annealing from 2.0 to 0.1 during Stage B.

The sampled retention ratio enters the selector through $g(\rho)$. The training ratios are $\{0.9, 0.5, 0.3, 0.1\}$, whereas evaluation covers seven retention ratios, $\{0.9, 0.7, 0.5, 0.4, 0.3, 0.2, 0.1\}$. All seven use the same trained checkpoint, including the ratios not sampled during training.

Table 7: Two-stage selector training. Retention is denoted by ρ; pruning rate is $p = 1 - \rho$. Approximate epoch counts follow the training configuration.

Setting	Stage A: initial alignment	Stage B: ratio conditioning
Data	Strong-signal subset	Source-weighted 779K mixture
Duration	Approximately 0.3 of an epoch	Approximately 2 epochs
Retention	Fixed $\rho = 0.9$	Four uniformly sampled steps from $\{0.5, 0.3, 0.1\}$, then one additional step at $\rho = 0.9$
STE temperature	Fixed $\beta = 2.0$	Cosine annealing, $2.0 \rightarrow 0.1$
Loss	Target-answer cross-entropy	Target-answer cross-entropy

Source-weighted data sampling. The instruction mixture assigns different sampling weights to task categories according to their visual-selection signal (Table 8). Fine-grained localization, OCR, chart understanding and grounding/referring tasks receive weight 2.5; general visual question answering and GPT-4V-generated visual question-answer or description data receive weight 1.0; generic descriptions, summaries and conversations in which the image is peripheral receive weight 0.5. Text-only samples without an image field are filtered by the data loader, and the selector retains a short-circuit path for zero visual tokens. These are sampling weights, not fixed proportions of the resulting training stream.

Table 8: Relative sampling weights by task category. Descriptions identify task types rather than an inventory of dataset names or splits.

Category	Task characteristics	Weight
Strong	Fine-grained localization; OCR and text recognition; chart understanding; grounding and referring tasks	2.5
Regular	General VQA; GPT-4V-generated visual questions, answers and descriptions	1.0
Weak	Generic descriptions and summaries; dialogue with peripheral visual content	0.5
Excluded	Text-only samples without an image field	0

Optimization. Optimization uses AdamW with learning rate 2×10^{-4}, weight decay 0.01, 1000 learning-rate warm-up steps, cosine decay and gradient clipping at 1.0. Positional embeddings and learnable scalars use a $0.1\times$ learning rate. Training uses an effective batch size of 64–128, bfloat16, eager attention and DDP on two 96 GB GPUs. The learning-rate warm-up and Stage A specify different aspects of the training schedule.

How the reported entries were produced. All main comparison figures and tables use this trained checkpoint, evaluated on the three remaining backbones without retraining. Reproducing the displayed SheafPrune envelope requires evaluating that checkpoint over $\delta \in \{0, 0.5, 1, 2\}$ and reproducing the original per-point selection rule. The aggregate tables alone do not establish that this choice used a held-out validation set, so no held-out selection or tuning-free deployment claim is made. A fixed deployment curve would instead use one precommitted δ across the complete grid. The fixed-configuration component study uses default $\delta = 1$ except in the no-correction arm.

Scope of the training evidence. The method is trained with target-answer cross-entropy through the frozen backbone. The uniform-score ablation changes the initialization supplied to the graph operator; it is not a comparison between trained and untrained checkpoints. The current component results therefore do not independently establish the necessity of every trainable submodule, nor do they establish that training is unnecessary.

N Complete Results

Table 9 contains the complete reported main-comparison grid: for each backbone it lists the evaluated methods and pruning rates, original scores on the ten benchmarks, the ten-benchmark mean and macro-average retention, and — at the three profiled budgets — prefill latency, latency reduction, decoding throughput and end-to-end FLOPs reduction. Table 1 of Section 4.2 is its $p = 0.7$ slice. Retention is computed per benchmark and then averaged across benchmarks, so each benchmark contributes on the same footing despite different native score scales; raw MME scores remain on their native scale in the MME column. Empty entries reflect either the implemented staged schedule's budget floor or combinations not evaluated here, as detailed in Appendix L. Missing values are not treated as zero. SheafPrune entries follow the per-point best-δ reporting protocol in Appendix M.

Table 9: **Complete results.** Every backbone, method and pruning rate in one table. Benchmark columns are the original scores of the final result workbook: Chart, Doc, Text, MMB and VQA2 are ChartQA (relaxed accuracy), DocVQA (ANLS), TextVQA, MMBench-EN and VQAv2; POPE is F1, MME keeps its native scale, the remaining benchmarks use accuracy-based metrics, and PruMerge is LLaVA-PruMerge. "Avg" is the ten-benchmark mean with MME rescaled as score$/28$; "RA" is the ten-benchmark macro-average retention (%) relative to the unpruned model of the same backbone, i.e. each benchmark is normalized first and the ratios are then averaged, which is the quantity every claim in the paper is stated in. "lat" is the measured prefill latency (ms) of the complete method — the forward pass plus that method's own selection cost — and "LR" the latency reduction (%) it gives against the unpruned prefill of the same backbone; "tok/s" is greedy decoding throughput and "FLOPs" the end-to-end FLOPs reduction (%) including the frozen vision tower. Efficiency was profiled at $p \in \{0.5, 0.7, 0.9\}$ on one 60-sample batch shared by all four backbones, so it is a per-backbone quantity and "–" in those columns marks a budget that was not profiled. SparseVLM and PyramidDrop prune inside the decoder in stages and their layer-averaged retention cannot fall below $1/S = 25\%$, so they have no $p \in \{0.8, 0.9\}$ entry at all; FasterVLM, VisionZip and LLaVA-PruMerge are reported only on LLaVA-1.5 in this evaluation; other backbone integrations are not evaluated here. Best pruning method per backbone, pruning rate and column in bold.

Method	p	General VQA		MLLM benchmarks			Text-oriented VQA			High-resolution		Aggregate		Efficiency				
	(%)	GQA	VQA2	MME	POPE	MMB	Chart	Doc	Text	HR-4K	HR-8K	Avg	RA	lat	×	LR	tok/s	FLOPs
LLaVA-NeXT-8B																		
No prune	0	65.0	82.1	1074.21	83.6	77.0	69.7	72.6	66.7	51.2	46.1	65.2	–	149.4	–	–	66.5	0.0
Random	10	64.9	82.4	1076.44	83.0	**77.2**	67.8	70.8	66.3	51.5	46.4	64.9	99.5	–	–	–	–	–
Random	30	64.5	82.1	1056.92	82.9	76.8	62.4	62.1	64.0	50.5	45.2	62.8	96.4	–	–	–	–	–
Random	50	64.4	80.0	1075.43	82.2	**77.3**	56.5	51.3	57.0	47.1	44.6	59.9	92.1	–	–	–	–	–
Random	60	63.9	80.0	1070.32	81.7	76.9	49.6	46.6	54.5	48.6	42.5	58.2	89.7	–	–	–	–	–
Random	70	63.8	79.3	1050.99	80.7	**77.2**	43.6	39.9	49.9	46.0	42.2	56.0	86.3	–	–	–	–	–
Random	80	63.1	77.7	1045.53	78.3	74.6	37.2	31.1	42.5	45.8	40.1	52.8	81.6	–	–	–	–	–
Random	90	59.1	72.5	1034.32	73.9	73.0	27.9	22.2	31.4	40.6	36.0	47.4	73.4	–	–	–	–	–
ToMe	10	65.0	82.6	1073.51	83.6	76.9	69.6	72.4	66.4	**51.9**	46.5	**65.3**	**100.2**	–	–	–	–	–
ToMe	30	65.0	82.1	1055.83	83.5	76.4	69.1	69.8	66.0	51.2	45.8	64.6	99.1	–	–	–	–	–
ToMe	50	**65.7**	81.6	1064.97	83.4	76.9	57.4	57.1	63.0	50.0	42.1	61.5	94.4	**78.6**	**1.90**	**47.4**	66.2	48.1
ToMe	60	**65.4**	81.8	1053.85	82.3	77.1	57.0	56.6	62.5	50.6	42.9	61.4	94.2	–	–	–	–	–
ToMe	70	64.5	80.8	1057.58	82.4	76.3	52.1	48.7	60.5	50.0	42.5	59.6	91.6	**52.6**	**2.84**	**64.8**	68.1	66.3
ToMe	80	63.9	79.3	1057.82	80.3	75.4	38.8	39.3	51.5	49.9	41.6	55.8	86.2	–	–	–	–	–
ToMe	90	62.5	75.9	**1043.83**	77.7	73.0	27.5	28.6	38.8	44.1	39.4	50.5	78.2	32.6	4.58	78.2	**68.8**	84.0
FastV	10	**65.2**	**82.6**	1057.88	83.6	76.9	69.6	72.6	66.5	51.6	46.4	65.3	100.0	–	–	–	–	–
FastV	30	64.7	**82.6**	1061.31	83.2	77.0	69.0	70.1	66.4	50.8	46.5	64.8	99.3	–	–	–	–	–
FastV	50	64.5	82.3	1054.56	83.4	77.0	67.1	66.7	65.2	51.5	47.0	64.2	98.6	80.8	1.85	45.9	52.0	49.0
FastV	60	64.0	82.1	1058.83	83.2	76.5	63.9	63.5	64.5	50.5	46.6	63.2	97.1	–	–	–	–	–
FastV	70	63.5	81.0	1056.34	82.5	76.1	59.2	55.2	61.7	51.4	45.5	61.4	94.5	55.7	2.68	62.7	51.7	67.5
FastV	80	61.7	78.1	1036.65	80.8	75.1	48.8	42.2	56.5	48.6	44.0	57.3	88.4	–	–	–	–	–
FastV	90	53.4	70.0	953.06	65.5	69.7	23.9	18.4	38.9	41.2	39.6	45.5	70.9	33.0	4.53	77.9	51.4	**85.4**

Table 9 (continued).

Method	p	General VQA		MLLM benchmarks			Text-oriented VQA			High-resolution		Aggregate		Efficiency				
	(%)	GQA	VQA2	MME	POPE	MMB	Chart	Doc	Text	HR-4K	HR-8K	Avg	RA	lat	×	LR	tok/s	FLOPs
SparseVLM	10	64.7	82.4	1063.02	**83.8**	77.2	69.5	72.2	66.4	51.6	**46.6**	65.2	100.0	–	–	–	–	–
SparseVLM	30	64.1	82.3	1059.00	83.8	76.7	69.4	72.5	66.3	**51.6**	46.6	65.1	99.8	–	–	–	–	–
SparseVLM	50	64.4	82.0	1058.22	**84.5**	76.6	68.1	71.0	64.1	**52.0**	46.9	64.7	99.3	84.4	1.77	43.5	51.6	49.7
SparseVLM	60	62.8	81.1	1055.18	**84.2**	77.0	64.0	65.4	59.8	**52.5**	**47.4**	63.2	97.1	–	–	–	–	–
SparseVLM	70	58.5	76.5	1039.67	83.0	76.8	45.0	46.4	46.9	51.0	46.1	56.7	87.8	63.3	2.36	57.6	51.2	67.9
SparseVLM	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SparseVLM	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	10	65.1	82.3	1062.25	83.6	77.0	69.7	71.9	66.4	51.4	46.5	65.2	99.9	–	–	–	–	–
PyramidDrop	30	**65.1**	82.2	1060.00	**83.8**	**77.2**	69.5	72.2	66.4	51.2	**46.8**	**65.2**	100.0	–	–	–	–	–
PyramidDrop	50	64.3	**82.4**	1057.51	83.8	76.9	**69.6**	71.3	66.4	51.6	**47.0**	65.1	99.8	84.9	1.76	43.2	50.9	**50.2**
PyramidDrop	60	64.2	**82.3**	1054.05	83.8	**77.2**	67.4	68.2	64.8	52.4	46.8	64.5	98.9	–	–	–	–	–
PyramidDrop	70	62.2	80.8	1057.45	**85.0**	76.1	45.4	49.3	55.5	**52.1**	45.6	59.0	91.0	63.0	2.37	57.8	50.9	**68.0**
PyramidDrop	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
G-Prune	10	64.7	82.0	**1085.27**	83.4	76.9	69.6	72.5	**67.1**	51.2	45.9	65.2	100.0	–	–	–	–	–
G-Prune	30	64.5	82.1	1071.84	83.3	77.0	**69.7**	71.7	**67.2**	50.1	46.2	65.0	99.6	–	–	–	–	–
G-Prune	50	64.9	82.2	**1083.51**	83.2	76.3	68.9	70.4	66.2	50.1	46.5	64.7	99.3	79.5	1.88	46.8	66.4	48.1
G-Prune	60	65.1	82.0	**1077.27**	83.8	75.8	68.0	68.6	66.1	51.4	46.2	64.6	99.1	–	–	–	–	–
G-Prune	70	**64.9**	81.6	1059.47	83.6	75.7	66.5	64.8	65.6	51.2	45.8	63.8	97.9	52.8	2.83	64.7	67.0	66.3
G-Prune	80	63.8	**80.9**	1055.26	83.5	75.6	63.4	59.0	62.7	50.6	45.6	62.3	95.7	–	–	–	–	–
G-Prune	90	62.1	76.7	1015.24	79.8	73.9	52.7	43.0	59.1	48.5	43.9	57.6	88.8	32.3	4.62	78.4	67.0	84.0
SheafPrune	10	65.0	82.1	1074.21	83.6	77.0	**69.7**	**72.6**	66.7	51.2	46.1	65.2	100.0	–	–	–	–	–
SheafPrune	30	65.0	82.1	**1074.19**	83.6	77.0	69.7	**72.6**	66.7	51.2	46.1	65.2	**100.0**	–	–	–	–	–
SheafPrune	50	65.0	82.1	1074.21	83.6	76.9	69.5	**72.2**	**66.7**	51.2	46.1	**65.2**	**99.9**	81.6	1.83	45.4	**67.9**	48.1
SheafPrune	60	65.0	82.1	1074.21	83.6	76.6	**69.0**	**70.8**	**66.6**	51.2	46.0	**64.9**	**99.5**	–	–	–	–	–
SheafPrune	70	64.8	**82.1**	**1074.21**	83.6	76.2	**67.5**	**67.2**	**66.1**	51.2	**46.1**	**64.3**	**98.7**	55.3	2.70	63.0	**68.3**	66.4
SheafPrune	80	**64.1**	80.8	**1065.10**	**83.6**	**76.3**	**64.6**	**62.2**	**64.3**	**51.0**	**46.1**	**63.1**	**97.0**	–	–	–	–	–
SheafPrune	90	**62.7**	**78.0**	1022.89	**80.8**	**74.2**	**54.1**	**46.6**	**60.0**	**49.7**	**44.1**	**58.7**	**90.4**	**29.0**	**5.15**	**80.6**	67.3	84.0
Qwen2.5-VL-7B																		
No prune	0	58.7	83.1	1242.65	84.3	83.3	87.8	94.2	84.3	71.4	63.9	75.5	–	113.6	–	–	71.0	0.0
Random	10	58.5	83.0	1238.03	**84.5**	83.1	86.8	93.9	84.2	70.9	63.6	75.3	99.7	–	–	–	–	–
Random	30	58.3	82.7	1231.26	83.7	83.5	84.1	93.0	81.9	69.6	62.5	74.3	98.5	–	–	–	–	–
Random	50	57.9	81.5	1229.30	82.9	81.8	77.8	88.3	78.1	69.8	60.2	72.2	95.9	–	–	–	–	–
Random	60	57.7	80.6	1205.57	83.0	**83.2**	71.3	85.4	75.2	66.5	59.4	70.5	93.8	–	–	–	–	–
Random	70	56.7	78.1	1178.81	82.3	**82.3**	64.7	77.3	71.0	65.2	57.4	67.7	90.3	–	–	–	–	–
Random	80	55.7	76.2	1169.27	80.3	**80.3**	52.6	63.6	64.6	63.4	53.6	63.2	84.9	–	–	–	–	–
Random	90	51.9	71.1	1092.10	74.1	77.9	37.7	44.5	49.2	58.6	51.9	55.6	75.5	–	–	–	–	–
ToMe	10	58.7	83.3	1241.65	84.3	83.4	87.6	94.1	84.2	71.6	64.1	75.6	100.1	–	–	–	–	–
ToMe	30	58.4	**83.2**	1238.00	83.9	83.3	86.4	93.6	83.5	70.9	64.2	75.2	99.6	–	–	–	–	–
ToMe	50	**58.8**	**82.1**	**1236.98**	83.0	82.7	82.9	93.2	82.4	69.1	62.6	74.1	98.2	**62.1**	**1.83**	**45.4**	67.0	35.8
ToMe	60	**58.8**	**81.5**	**1224.74**	83.9	82.4	79.3	90.7	81.3	67.9	63.6	73.3	97.3	–	–	–	–	–
ToMe	70	**58.2**	**80.8**	**1214.42**	83.3	81.3	73.1	84.7	79.5	68.1	60.9	71.3	94.9	**43.4**	**2.62**	**61.8**	68.1	49.5
ToMe	80	**56.7**	**78.7**	**1185.98**	82.4	80.1	62.4	75.0	74.1	67.2	59.6	67.9	90.7	–	–	–	–	–
ToMe	90	**54.7**	**74.8**	**1131.61**	79.6	**78.9**	48.1	51.3	**61.9**	65.9	57.8	61.3	82.9	35.8	3.17	68.5	69.9	63.0
FastV	10	58.5	83.2	1233.22	84.1	83.2	87.6	**94.4**	84.1	71.6	**64.4**	75.5	100.0	–	–	–	–	–
FastV	30	57.9	82.7	1241.82	84.5	82.5	86.1	92.9	84.0	**72.4**	**65.4**	75.3	99.7	–	–	–	–	–
FastV	50	56.9	81.5	1236.19	84.2	82.5	80.6	90.1	83.2	72.2	63.5	73.9	98.0	64.5	1.76	43.2	46.4	35.9
FastV	60	54.7	79.5	1218.42	82.0	81.9	74.5	84.4	82.9	70.9	63.8	71.8	95.4	–	–	–	–	–
FastV	70	51.7	76.8	1170.01	79.9	79.5	63.0	74.5	**82.3**	68.1	61.9	68.0	90.6	47.9	2.37	57.8	46.4	49.7
FastV	80	47.7	71.5	1106.28	74.9	74.5	40.4	55.2	**76.6**	63.5	57.9	60.2	81.0	–	–	–	–	–
FastV	90	38.0	44.8	869.22	18.1	56.9	15.9	25.4	48.1	53.6	45.4	37.7	52.7	30.4	3.74	73.3	46.1	**63.1**
SparseVLM	10	58.7	83.1	1244.32	84.5	83.0	87.3	94.0	84.2	**72.0**	64.0	75.5	100.0	–	–	–	–	–
SparseVLM	30	58.1	82.4	1241.01	84.9	**83.5**	85.0	93.9	83.6	71.9	64.2	75.2	99.6	–	–	–	–	–
SparseVLM	50	56.5	80.3	1229.35	83.9	**82.9**	77.7	92.3	83.7	71.9	64.5	73.8	97.8	72.8	1.56	35.9	45.9	36.7
SparseVLM	60	54.7	76.5	1221.25	83.1	80.6	65.5	87.3	82.0	**72.8**	64.2	71.0	94.6	–	–	–	–	–
SparseVLM	70	49.2	68.9	1124.35	76.2	76.9	27.3	67.3	66.7	69.5	63.0	60.5	81.8	57.1	1.99	49.7	46.0	50.3
SparseVLM	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SparseVLM	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	10	58.5	83.3	1241.84	84.1	82.8	87.5	94.0	84.2	71.4	64.2	75.4	99.9	–	–	–	–	–
PyramidDrop	30	58.4	82.8	**1243.73**	85.0	83.0	86.7	94.2	83.7	72.0	64.4	75.5	100.0	–	–	–	–	–
PyramidDrop	50	56.9	81.9	1215.19	**84.6**	81.0	81.7	92.3	**84.4**	71.6	64.8	74.3	98.4	71.9	1.58	36.7	45.6	**37.1**
PyramidDrop	60	54.5	79.8	1166.81	82.2	78.3	73.9	88.9	**83.5**	70.5	64.0	71.7	95.1	–	–	–	–	–
PyramidDrop	70	48.1	71.5	1018.31	74.9	71.3	41.0	67.6	69.1	63.2	54.6	59.8	79.9	56.8	2.00	50.0	45.7	**50.4**
PyramidDrop	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
G-Prune	10	**59.1**	**83.3**	**1245.59**	84.3	**83.4**	87.7	94.1	**84.3**	71.5	63.9	**75.6**	**100.1**	–	–	–	–	–
G-Prune	30	57.9	82.3	1242.56	**85.0**	83.3	86.4	**94.3**	84.2	72.0	63.9	75.4	99.8	–	–	–	–	–
G-Prune	50	55.7	80.2	1213.94	84.5	81.8	82.5	93.4	84.2	**72.8**	**65.2**	74.3	98.4	63.5	1.79	44.1	69.2	35.7
G-Prune	60	53.9	78.5	1184.42	83.5	80.6	77.0	91.6	81.7	72.6	**64.4**	72.6	96.2	–	–	–	–	–
G-Prune	70	51.2	74.8	1115.52	80.1	78.5	66.8	88.0	78.4	**72.8**	**64.5**	69.5	92.2	44.0	2.58	61.2	67.5	49.5
G-Prune	80	48.9	68.8	1011.88	73.5	76.0	51.3	76.2	71.4	**70.8**	**63.5**	63.7	84.8	–	–	–	–	–
G-Prune	90	43.5	60.1	928.41	61.8	67.2	28.7	52.6	52.0	68.0	60.0	52.7	71.4	30.1	3.78	73.5	67.2	62.9
SheafPrune	10	58.6	83.1	1242.65	84.2	83.3	**87.8**	94.2	84.3	71.4	63.9	75.5	100.0	–	–	–	–	–
SheafPrune	30	**58.7**	83.1	1242.52	84.3	83.2	**87.7**	94.2	**84.3**	71.4	63.9	**75.5**	**100.0**	–	–	–	–	–
SheafPrune	50	58.7	81.9	1223.50	84.3	82.3	**84.2**	**93.4**	83.4	71.4	63.8	**74.7**	**99.0**	65.7	1.73	42.2	**72.0**	35.8
SheafPrune	60	58.1	80.8	1204.79	**84.3**	82.6	**81.9**	**91.7**	82.0	71.4	63.8	**74.0**	**98.0**	–	–	–	–	–
SheafPrune	70	57.0	80.0	1156.14	**84.2**	81.2	**77.1**	**88.1**	79.1	71.4	63.5	**72.3**	**95.8**	45.8	2.48	59.7	**72.2**	49.6
SheafPrune	80	55.5	78.0	1111.07	**83.4**	79.7	**69.6**	**79.0**	73.8	69.8	63.3	**69.2**	**92.0**	–	–	–	–	–
SheafPrune	90	53.5	74.6	1082.23	**81.1**	77.2	**53.6**	**62.7**	59.4	**68.1**	**61.1**	**63.0**	**84.6**	**29.4**	**3.87**	**74.2**	**71.4**	63.0
LLaVA-1.5-7B																		
No prune	0	58.7	75.8	1030.02	81.1	67.9	18.0	24.6	47.4	33.9	36.2	48.1	–	39.7	–	–	72.3	0.0
Random	10	58.6	76.0	1031.50	80.1	68.4	17.7	23.7	47.1	33.6	36.2	47.8	99.2	–	–	–	–	–
Random	30	58.4	75.2	1026.39	78.9	67.4	17.4	21.3	43.6	33.4	35.5	46.8	96.7	–	–	–	–	–
Random	50	58.3	74.7	1029.63	79.0	67.4	16.1	19.7	40.9	33.8	35.4	46.2	94.7	–	–	–	–	–
Random	60	57.9	73.8	**1029.04**	77.5	66.8	15.5	17.3	39.3	33.8	36.0	45.4	92.8	–	–	–	–	–
Random	70	56.8	73.3	1004.01	77.3	65.6	14.3	15.8	35.3	33.2	35.0	44.2	89.5	–	–	–	–	–

Table 9 (continued).

Method	p	General VQA		MLLM benchmarks			Text-oriented VQA			High-resolution		Aggregate		Efficiency				
	(%)	GQA	VQA2	MME	POPE	MMB	Chart	Doc	Text	HR-4K	HR-8K	Avg	RA	lat	×	LR	tok/s	FLOPs
Random	80	55.5	68.8	969.01	75.8	64.7	13.5	14.8	30.3	32.2	35.0	42.5	85.8	–	–	–	–	–
Random	90	51.5	62.5	890.70	73.5	61.2	11.9	10.6	22.9	30.9	32.9	38.9	77.6	–	–	–	–	–
ToMe	10	58.7	76.0	1032.26	81.0	67.8	17.8	24.9	46.8	33.8	**36.6**	48.0	99.9	–	–	–	–	–
ToMe	30	58.5	74.8	1029.22	80.5	68.0	18.0	23.8	47.2	35.0	35.0	47.7	99.3	–	–	–	–	–
ToMe	50	58.5	74.5	1033.46	79.8	67.2	16.0	18.8	44.4	35.0	36.0	46.7	95.7	24.7	1.61	37.9	72.6	45.8
ToMe	60	58.0	74.7	1025.11	79.3	66.6	15.8	19.3	44.4	35.1	36.6	46.6	95.7	–	–	–	–	–
ToMe	70	57.4	73.9	1006.63	**79.7**	66.2	16.0	17.8	42.7	34.4	35.6	46.0	93.9	19.6	2.03	50.7	72.0	63.9
ToMe	80	**57.3**	72.5	997.93	**78.6**	64.9	14.5	15.0	35.7	**35.9**	33.1	44.3	89.6	–	–	–	–	–
ToMe	90	**53.9**	67.2	941.83	**76.2**	60.9	13.2	12.0	27.2	31.9	32.4	40.9	81.8	18.0	2.21	54.8	72.8	81.7
FastV	10	58.5	75.9	**1036.93**	80.9	68.0	**18.2**	24.6	**47.6**	33.9	35.5	48.0	100.0	–	–	–	–	–
FastV	30	58.1	74.9	1024.33	78.6	67.6	17.6	23.6	47.7	33.8	36.4	47.5	98.7	–	–	–	–	–
FastV	50	56.8	73.8	1013.15	77.3	67.1	17.1	21.6	45.2	33.1	36.9	46.5	96.4	**24.4**	**1.63**	**38.7**	57.7	45.8
FastV	60	56.2	74.0	998.77	75.8	67.0	16.6	21.4	44.9	33.4	36.2	46.1	95.5	–	–	–	–	–
FastV	70	53.9	71.8	983.38	73.8	66.5	15.8	18.3	42.2	33.6	36.1	44.7	92.1	19.7	2.02	50.5	59.6	63.9
FastV	80	52.1	69.3	942.30	65.7	65.1	14.8	14.5	38.2	32.8	35.2	42.1	86.4	–	–	–	–	–
FastV	90	44.0	54.2	810.13	27.6	54.0	13.0	9.4	23.7	30.1	31.4	31.6	67.5	**17.3**	**2.30**	**56.5**	59.9	81.8
SparseVLM	10	**59.1**	75.6	1034.65	80.9	67.8	18.2	24.4	47.3	33.6	35.9	48.0	99.8	–	–	–	–	–
SparseVLM	30	57.5	75.7	1033.31	**81.5**	68.0	18.1	24.7	46.8	33.5	35.2	47.8	99.4	–	–	–	–	–
SparseVLM	50	56.9	74.7	**1035.86**	**81.3**	**67.9**	17.4	22.8	43.8	33.4	35.8	47.1	97.5	29.2	1.36	26.5	58.2	46.1
SparseVLM	60	56.2	72.3	1018.46	78.1	67.5	15.9	18.2	37.1	34.4	34.8	45.1	92.3	–	–	–	–	–
SparseVLM	70	49.8	64.6	972.47	72.9	65.6	13.7	10.6	21.8	35.2	34.1	40.3	81.4	27.2	1.46	31.5	58.5	64.0
SparseVLM	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SparseVLM	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	10	58.4	75.7	1034.90	81.1	68.0	18.1	24.7	47.4	33.8	35.9	48.0	99.9	–	–	–	–	–
PyramidDrop	30	58.1	75.7	1029.69	80.8	**68.0**	**18.1**	24.7	**47.8**	33.4	35.5	47.9	99.6	–	–	–	–	–
PyramidDrop	50	57.8	75.8	1032.02	79.8	67.5	**18.1**	24.3	**47.5**	33.4	36.2	47.7	99.4	29.2	1.36	26.5	58.4	**46.2**
PyramidDrop	60	57.4	74.6	1022.74	78.4	67.4	17.5	23.4	46.5	34.8	34.5	47.1	97.9	–	–	–	–	–
PyramidDrop	70	55.3	72.1	977.35	73.5	65.6	16.0	17.1	40.4	33.1	34.4	44.2	90.7	26.5	1.50	33.3	58.5	**64.1**
PyramidDrop	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
G-Prune	10	58.2	**76.1**	1025.32	80.7	**68.5**	18.0	24.6	47.3	33.4	35.1	47.9	99.4	–	–	–	–	–
G-Prune	30	57.7	75.3	1015.39	79.7	66.9	17.5	24.7	47.7	35.1	35.8	47.7	99.3	–	–	–	–	–
G-Prune	50	56.6	73.9	1017.94	77.9	67.3	17.4	24.4	47.2	34.1	36.4	47.2	98.4	24.7	1.61	37.9	73.6	45.8
G-Prune	60	56.7	73.0	1021.16	76.9	66.7	17.6	24.0	46.7	34.0	36.2	46.8	97.9	–	–	–	–	–
G-Prune	70	55.1	72.9	1013.33	75.5	65.2	17.1	22.6	46.5	33.1	36.1	46.0	96.0	**19.0**	**2.09**	**52.2**	73.6	63.9
G-Prune	80	52.7	71.0	978.89	71.5	64.6	16.4	22.0	44.9	33.9	35.6	44.8	93.5	–	–	–	–	–
G-Prune	90	47.3	62.8	865.14	52.7	57.9	14.6	16.8	40.2	33.2	34.2	39.1	82.4	18.1	2.19	54.3	74.8	81.7
FasterVLM	10	58.8	75.9	1025.69	81.4	68.0	18.0	24.9	47.2	33.6	36.0	48.0	99.9	–	–	–	–	–
FasterVLM	30	**58.9**	**76.1**	1034.64	80.9	67.3	17.4	25.2	46.7	35.0	37.1	48.1	100.3	–	–	–	–	–
FasterVLM	50	58.5	**76.0**	1024.47	80.1	67.4	17.4	**24.8**	46.5	35.6	36.0	47.9	99.7	29.4	1.35	25.9	73.6	45.8
FasterVLM	60	58.2	75.0	1022.66	78.2	67.5	17.4	23.7	46.1	**35.9**	36.2	47.5	98.9	–	–	–	–	–
FasterVLM	70	55.9	74.3	1023.51	76.4	66.8	17.6	22.7	45.5	**35.8**	37.1	46.9	97.9	23.8	1.67	40.1	**73.9**	63.9
FasterVLM	80	55.2	72.0	1003.94	73.2	66.0	17.0	22.0	44.9	34.9	36.2	45.7	95.5	–	–	–	–	–
FasterVLM	90	51.7	68.9	**974.25**	65.5	64.9	16.1	18.3	41.7	**34.5**	**35.5**	43.2	**90.1**	21.5	1.85	45.9	**74.8**	81.7
VisionZip	10	58.7	75.9	1023.18	**81.4**	68.0	17.9	**25.1**	47.2	33.8	36.2	**48.1**	**100.0**	–	–	–	–	–
VisionZip	30	58.6	75.9	1034.44	80.9	67.3	17.3	**25.3**	46.7	**35.2**	**37.4**	**48.2**	**100.3**	–	–	–	–	–
VisionZip	50	**58.8**	75.6	1019.21	79.5	67.5	17.4	24.5	46.3	**35.6**	36.8	47.8	99.6	33.4	1.19	16.0	73.6	45.8
VisionZip	60	57.6	74.6	1024.62	79.0	67.2	17.7	23.5	46.0	35.4	36.6	47.4	98.8	–	–	–	–	–
VisionZip	70	56.5	74.4	**1025.46**	76.9	67.0	17.7	23.1	45.6	35.1	**37.1**	47.0	98.1	26.5	1.50	33.3	72.6	63.9
VisionZip	80	54.6	72.6	**1004.97**	74.5	66.1	16.8	21.6	44.4	34.1	35.1	45.6	94.8	–	–	–	–	–
VisionZip	90	51.9	69.5	969.49	68.0	**65.1**	**16.1**	17.3	40.8	34.4	34.4	43.2	89.6	22.4	1.77	43.5	74.3	81.7
PruMerge	10	58.7	75.9	1024.12	81.3	68.0	17.8	25.0	47.2	33.6	36.1	48.0	99.9	–	–	–	–	–
PruMerge	30	58.7	76.0	**1035.17**	80.5	67.3	17.1	25.0	47.0	35.0	37.0	48.1	100.0	–	–	–	–	–
PruMerge	50	58.6	75.6	1026.63	80.2	67.3	17.5	24.0	46.5	35.4	**37.2**	47.9	99.7	40.5	0.98	-2.0	**73.7**	45.8
PruMerge	60	57.7	75.1	1025.16	78.9	67.5	17.6	24.1	46.9	35.0	**37.5**	47.7	99.5	–	–	–	–	–
PruMerge	70	56.7	74.4	1023.79	77.7	66.7	17.0	22.5	45.8	34.8	36.8	46.9	97.4	34.5	1.15	13.0	72.9	63.9
PruMerge	80	54.8	**73.5**	1003.64	77.3	**66.6**	16.2	20.4	44.2	33.2	**36.2**	45.8	94.5	–	–	–	–	–
PruMerge	90	53.3	**70.1**	964.23	72.2	64.8	15.3	16.3	40.3	33.0	34.6	**43.4**	89.0	27.0	1.47	32.0	74.4	81.7
SheafPrune	10	58.7	75.8	1029.62	81.1	67.9	18.0	24.6	47.4	**33.9**	36.2	48.0	100.0	–	–	–	–	–
SheafPrune	30	58.7	75.8	1029.65	81.1	67.8	18.0	24.6	47.4	33.9	36.2	48.0	100.0	–	–	–	–	–
SheafPrune	50	58.6	75.7	1028.76	81.0	67.8	18.0	24.6	47.4	33.9	36.2	**48.0**	**99.9**	26.8	1.48	32.4	73.4	45.8
SheafPrune	60	**58.4**	**75.5**	1026.86	**80.7**	**67.8**	**18.0**	**24.6**	**47.4**	33.9	36.2	**47.9**	**99.8**	–	–	–	–	–
SheafPrune	70	**57.7**	**74.9**	1019.97	79.3	**67.3**	**17.9**	**24.2**	**47.1**	33.9	36.2	**47.5**	**99.0**	21.2	1.87	46.5	72.9	64.0
SheafPrune	80	56.2	73.2	995.12	76.5	66.1	**17.3**	**23.5**	**46.6**	33.6	35.9	**46.4**	**96.8**	–	–	–	–	–
SheafPrune	90	51.4	67.6	924.23	63.6	61.6	15.8	**19.7**	**42.6**	33.9	35.1	42.4	89.0	18.8	2.11	52.6	74.6	**81.9**
InternVL3-8B																		
No prune	0	61.3	79.3	1253.19	89.6	87.0	86.5	92.5	82.3	70.0	61.9	75.5	–	128.7	–	–	70.9	0.0
Random	10	61.2	79.1	1246.94	**89.9**	87.2	85.9	91.7	81.7	**70.2**	61.8	75.3	99.7	–	–	–	–	–
Random	30	61.4	79.1	1235.22	89.4	86.9	84.0	86.4	79.4	68.9	60.6	74.0	98.1	–	–	–	–	–
Random	50	**61.7**	78.8	1223.11	89.0	86.6	77.8	75.0	74.6	67.1	58.5	71.3	94.8	–	–	–	–	–
Random	60	60.9	78.6	1202.92	89.1	86.6	72.9	64.2	69.8	65.6	59.1	69.0	92.1	–	–	–	–	–
Random	70	61.0	77.6	1195.06	88.8	85.6	64.3	51.6	62.6	64.5	58.2	65.7	88.3	–	–	–	–	–
Random	80	59.3	74.9	1170.54	88.3	84.2	51.4	39.7	54.5	62.0	54.2	61.0	82.5	–	–	–	–	–
Random	90	**56.8**	**70.0**	1103.52	**85.6**	**80.0**	35.0	25.4	39.7	57.0	50.5	53.9	73.6	–	–	–	–	–
ToMe	10	61.1	79.4	1251.28	89.3	87.2	85.7	91.9	81.6	69.4	61.5	75.2	99.5	–	–	–	–	–
ToMe	30	61.3	**79.8**	1253.62	89.6	86.5	84.8	89.0	80.1	69.6	61.6	74.7	99.1	–	–	–	–	–
ToMe	50	60.3	79.2	1228.97	89.1	86.1	78.8	79.5	77.8	69.5	58.1	72.2	95.9	**71.5**	**1.80**	**44.4**	70.6	42.9
ToMe	60	60.5	**79.3**	1226.28	88.5	86.2	75.5	74.9	76.0	68.5	57.5	71.1	94.6	–	–	–	–	–
ToMe	70	60.5	77.8	1199.65	88.6	85.8	63.6	62.1	72.4	67.4	57.4	67.8	90.8	**46.3**	**2.78**	**64.0**	71.3	59.3
ToMe	80	59.7	75.2	1184.32	88.1	83.7	46.3	44.4	63.8	66.8	57.0	62.7	84.8	–	–	–	–	–
ToMe	90	55.3	69.2	1105.18	84.9	78.3	27.0	23.2	47.3	63.0	51.6	53.9	73.8	33.5	3.84	74.0	**72.0**	75.3
FastV	10	61.3	79.3	1250.62	89.6	87.1	86.4	92.1	82.2	69.9	**61.9**	75.5	99.9	–	–	–	–	–
FastV	30	61.2	79.3	1252.63	89.6	87.0	86.0	91.8	82.5	69.8	61.0	75.3	99.7	–	–	–	–	–
FastV	50	60.8	**79.5**	1252.67	88.6	**87.5**	84.8	89.2	**82.3**	70.6	61.6	75.0	99.4	74.8	1.72	41.9	54.8	43.0
FastV	60	60.9	79.2	1245.49	88.8	**87.2**	82.5	85.3	**81.6**	70.1	61.2	74.1	98.4	–	–	–	–	–
FastV	70	60.5	**79.1**	**1240.14**	88.7	**86.2**	76.5	75.5	**81.4**	70.4	61.5	72.4	96.4	52.3	2.46	59.3	54.7	59.4
FastV	80	59.5	**78.1**	**1218.23**	87.6	**84.9**	62.6	60.4	**75.6**	66.8	58.1	67.7	90.7	–	–	–	–	–
FastV	90	52.4	67.8	1093.68	79.8	72.0	18.7	21.2	45.6	50.9	45.2	49.3	67.6	34.2	3.76	73.4	54.2	75.3

Table 9 (continued).

Method	p	General VQA		MLLM benchmarks			Text-oriented VQA			High-resolution		Aggregate		Efficiency				
	(%)	GQA	VQA2	MME	POPE	MMB	Chart	Doc	Text	HR-4K	HR-8K	Avg	RA	lat	×	LR	tok/s	FLOPs
SparseVLM	10	61.4	**79.4**	1252.88	89.3	87.0	86.2	92.5	**82.5**	70.0	61.4	75.4	99.9	–	–	–	–	–
SparseVLM	30	**61.5**	79.1	1249.17	89.4	**87.4**	86.0	91.8	82.3	69.9	61.5	75.3	99.8	–	–	–	–	–
SparseVLM	50	61.0	78.6	**1252.75**	88.8	86.9	82.6	88.4	81.2	70.2	61.9	74.4	98.8	77.5	1.66	39.8	54.1	43.9
SparseVLM	60	59.9	78.2	**1245.86**	88.5	86.9	75.4	83.4	78.3	70.6	60.8	72.7	96.6	–	–	–	–	–
SparseVLM	70	58.3	76.4	1222.94	87.5	84.9	45.2	60.8	66.5	68.8	61.2	65.3	88.0	59.3	2.17	53.9	54.3	60.0
SparseVLM	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SparseVLM	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	10	**61.5**	78.9	1252.95	89.4	**87.3**	86.3	92.3	82.3	69.8	61.8	75.4	99.9	–	–	–	–	–
PyramidDrop	30	61.3	79.2	**1256.13**	89.3	87.0	86.3	92.2	82.4	70.1	61.1	75.4	99.8	–	–	–	–	–
PyramidDrop	50	61.1	78.7	1241.31	88.7	86.5	85.0	89.8	81.9	70.1	61.8	74.8	99.1	76.2	1.69	40.8	53.8	**44.4**
PyramidDrop	60	60.3	78.0	1230.00	88.5	85.9	81.5	85.9	80.9	70.0	62.8	73.8	97.9	–	–	–	–	–
PyramidDrop	70	58.7	76.2	1160.00	86.8	81.3	60.6	62.4	68.8	64.5	57.5	65.8	88.1	59.3	2.17	53.9	53.7	**60.1**
PyramidDrop	80	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PyramidDrop	90	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
G-Prune	10	61.3	79.3	1249.82	89.5	87.2	86.2	**92.9**	82.4	69.5	61.4	75.4	99.8	–	–	–	–	–
G-Prune	30	61.0	79.2	1246.37	**89.7**	86.5	85.7	91.8	**82.6**	**70.1**	61.5	75.3	99.7	–	–	–	–	–
G-Prune	50	61.3	79.3	1247.56	89.4	86.6	81.9	88.0	81.5	**70.9**	**62.9**	74.6	99.0	72.3	1.78	43.8	70.8	42.9
G-Prune	60	60.5	79.2	1242.41	89.2	86.1	78.2	84.3	81.0	**70.8**	**62.8**	73.6	97.8	–	–	–	–	–
G-Prune	70	59.7	78.0	1233.63	89.3	85.5	72.2	77.8	78.5	**71.0**	**63.2**	71.9	95.8	47.0	2.74	63.5	71.0	59.3
G-Prune	80	56.9	76.5	1188.04	88.0	83.3	57.0	66.0	72.7	**70.4**	**64.2**	67.8	90.8	–	–	–	–	–
G-Prune	90	52.9	69.3	**1106.53**	84.7	78.0	31.0	**42.8**	**56.8**	**70.2**	**59.4**	**58.5**	**79.4**	**30.2**	**4.26**	**76.5**	71.4	75.2
SheafPrune	10	61.3	79.3	**1252.99**	89.6	87.0	**86.5**	92.5	82.3	69.9	61.8	**75.5**	**100.0**	–	–	–	–	–
SheafPrune	30	61.3	79.3	1251.85	89.6	87.0	**86.4**	**92.3**	82.2	69.9	**61.7**	**75.4**	**99.9**	–	–	–	–	–
SheafPrune	50	61.3	79.2	1247.33	**89.5**	86.8	**85.9**	**91.0**	81.8	70.0	61.9	**75.2**	**99.6**	76.2	1.69	40.8	**71.9**	42.9
SheafPrune	60	**61.2**	79.0	1237.98	**89.5**	86.5	**84.5**	**88.4**	81.0	70.0	61.8	**74.6**	**98.9**	–	–	–	–	–
SheafPrune	70	**61.0**	78.0	1213.21	**89.5**	86.0	**81.6**	**83.3**	79.1	70.0	61.6	**73.4**	**97.3**	50.3	2.56	60.9	**72.3**	59.6
SheafPrune	80	**59.9**	75.8	1186.28	**89.2**	84.0	**71.2**	**69.8**	71.3	68.1	59.6	**69.1**	**92.2**	–	–	–	–	–
SheafPrune	90	54.6	67.3	1074.99	85.1	76.9	**41.2**	38.9	42.2	63.6	55.9	56.4	76.5	30.4	4.24	76.4	71.6	**75.4**

O Statistical Scope of the Final Results

The reported tables contain aggregate benchmark scores rather than the paired predictions and repeated-training results required for inferential comparisons. They support comparisons of the displayed point estimates, but not paired bootstrap tests, McNemar tests, seed-variance estimates or multiplicity-corrected significance claims. No universal accuracy "noise floor" is inferred from these aggregates. A future statistical comparison should fix the hyperparameter policy, retain the corresponding per-sample predictions and distinguish evaluation uncertainty from variation across training seeds.

P Component Ablation

Table 10 expands Table 2 with initialization variants and task groups. The full method improves over uniform initialization by 0.93, 1.25 and 1.20 RA points at $p = 0.5, 0.7, 0.9$, respectively. At $p = 0.9$, removing relevance ($\lambda = 0$) gives 86.40% RA and removing the visual prior ($\lambda = 1$) gives 85.00%, both below the uniform arm's 88.19% and the full method's 89.39%. These comparisons support the added value of combining relevance with the visual prior within the fixed graph operator, while uniform initialization remains a strong baseline. The Text-group difference between the full method and uniform initialization is 1.65 points.

Dense operator definition. The documented dense variant replaces the off-support logits by zero before row softmax. With $M_{ij} = \mathbb{1}[j \in \mathcal{N}(i)]$ on valid pairs, its mathematical form is

$$T_{ij}^{\text{dense}} = \frac{\exp(M_{ij}\,\text{sim}_{ij})}{\sum_{l:v_l=1}\exp(M_{il}\,\text{sim}_{il})}. \tag{42}$$

Every valid non-edge then has positive weight because $\exp(0) = 1$. This changes both propagation support and normalization; it is not a denominator-only intervention. The reported ablation keeps the threshold-graph degree correction at $\delta = 1$. It is an internal SheafPrune intervention, not a claim that official G-Prune omits masking.

Table 11 reports a separate norm-plus-graph family on LLaVA-1.5. Its uniform and norm initializations differ by at most 0.1 points with propagation, but these are not SheafPrune arms. They provide a related observation rather than a replication of the present ablation.

Table 10: **Complete component ablation of SheafPrune on LLaVA-NeXT-8B.** The default configuration matches Table 2; "Text" averages ChartQA, DocVQA and TextVQA, "OCR" adds HR-Bench 4K and 8K, and "Gen." averages GQA, VQAv2, MMBench-EN, POPE and MME. Rows marked † give additional initialization variants.

	Configuration				Ten-benchmark RA (%)			At p=0.9		
Arm	$s^{(0)}$	Prop.	Norm.	δ	0.5	0.7	0.9	Text	OCR	Gen.
Random selection	—	×	—	—	92.12	86.32	73.38	39.20	54.98	91.77
Degree only	uniform	×	—	1	95.77	92.14	81.38	57.09	69.03	93.74
Visual prior only	$\overline{n}$	×	—	0	91.13	84.40	70.23	34.10	49.88	90.57
Relevance only†	$\widehat{r}$	×	—	0	91.81	85.80	72.58	38.00	53.78	91.37
Uniform initial score	uniform	✓	support	1	98.50	96.72	88.19	73.32	81.54	94.63
No relevance (λ=0)†	$\overline{n}$	✓	support	1	98.49	96.11	86.40	69.17	77.28	95.53
No visual prior (λ=1)†	$\widehat{r}$	✓	support	1	98.35	95.19	85.00	66.97	75.08	94.93
No degree correction	both	✓	support	0	91.88	85.97	72.80	38.30	54.08	91.52
Dense normalization	both	✓	dense	1	91.74	85.78	72.68	38.10	53.88	91.47
SheafPrune	both	✓	support	1	**99.43**	**97.97**	**89.39**	**74.97**	**83.08**	**95.71**

Table 11: Component ablation of the *norm-plus-graph baseline family* on LLaVA-1.5-7B: accuracy retention (%) averaged over ten benchmarks. These are arms of that family, not of SheafPrune. B = random selection, N = feature norm without a graph, U = uniform initialization with diffusion, F = norm initialization with diffusion.

Configuration	ρ=0.9	0.7	0.5	0.4	0.3	0.2	0.1
B: no norm, no graph	99.24	96.66	94.71	92.75	89.54	85.85	77.60
N: norm, no graph	99.52	97.00	89.38	83.23	74.62	59.48	53.72
U: uniform, graph	99.41	99.38	98.40	97.88	96.06	93.47	82.39
F: norm, graph	99.43	99.30	98.43	97.88	95.96	93.48	82.39
F − U (value of the norm prior)	+0.02	−0.07	+0.03	−0.00	−0.10	+0.01	−0.00
U − B (value of the diffusion)	+0.17	+2.71	+3.69	+5.12	+6.52	+7.62	+4.79

P.1 Hyperparameter Analysis

The component study above already covers the remaining budgets, so it is not repeated here; Table 12 reports the remaining two sensitivity studies.

Table 12: Sensitivity of the propagation depth t and of the similarity threshold s, measured as the mean score over four benchmarks at $\downarrow 90\%$ on Qwen2.5-VL. These are point estimates, without a measured noise threshold.

(a) Propagation depth		*(b) Similarity threshold*	
t	Mean	s	Mean
1	54.06	0.3	49.11
3	**54.23**	0.4	53.61
5	53.85	0.5	**53.85**
10	53.85	0.6	51.18
20	53.06	0.7	44.90

Graph construction and hyperparameters. For $t \in \{1, 3, 5, 10\}$ the reported mean scores range from 53.85 to 54.23, a spread of 0.38 points; $t = 20$ gives 53.06. This describes the tested configuration and does not establish a noise threshold or eliminate the need for depth selection elsewhere. The separate threshold sweep uses an absolute cosine threshold s, rather than the default retained-relation fraction $q_v = 0.05$; its largest displayed mean is 53.85 at $s = 0.5$. The two sweeps should not be interpreted as one common configuration grid without their run settings.

The existing k-NN comparison reports a 23–24-point decrease in the DocVQA score for $k \in \{16, 32, 64, 128\}$. It changes neighborhood construction and may change symmetry and degree

distribution together. It therefore supports sensitivity to the tested graph construction, without isolating the effect of degree information. Complete per-k run scores and the symmetrization convention are needed for a more specific attribution.

Q Spatial Coverage Diagnosis

This section states how the diagnosis of Section 4.3 is measured and reports the coverage figures it is based on (Table 13). Retained tokens are mapped back onto the image grid, and two quantities are recorded on the original visual-token layout so that benchmarks with different visual sequence lengths remain comparable: grid coverage, the fraction of 8×8 cells that keep at least one token, and the mean nearest-neighbor distance between retained tokens. The retained subsets show similar spatial concentration for both graph-based selectors: at $\rho = 0.1$ on GQA the coverage is 0.417 for SheafPrune and 0.385 for G-Prune, against 0.500 for random selection. An explicit coverage constraint is a possible future extension, but its effect is not tested here. These cross-task observations do not isolate coverage as the cause of accuracy differences, because token counts, image content and tokenization also change.

Table 13: Grid coverage at $\downarrow 90\%$: fraction of 8×8 cells that keep a token.

	N_v	Random	Ours
GQA	345	0.500	0.417
VQAv2	345	0.432	0.333
DocVQA	4032	0.859	0.771
HR-Bench	4096	1.000	0.844

R Additional Qualitative Samples

Figure 5 repeats the protocol of Figure 1 on four further samples, two from TextVQA and two from VQAv2, shown at the same three budgets.

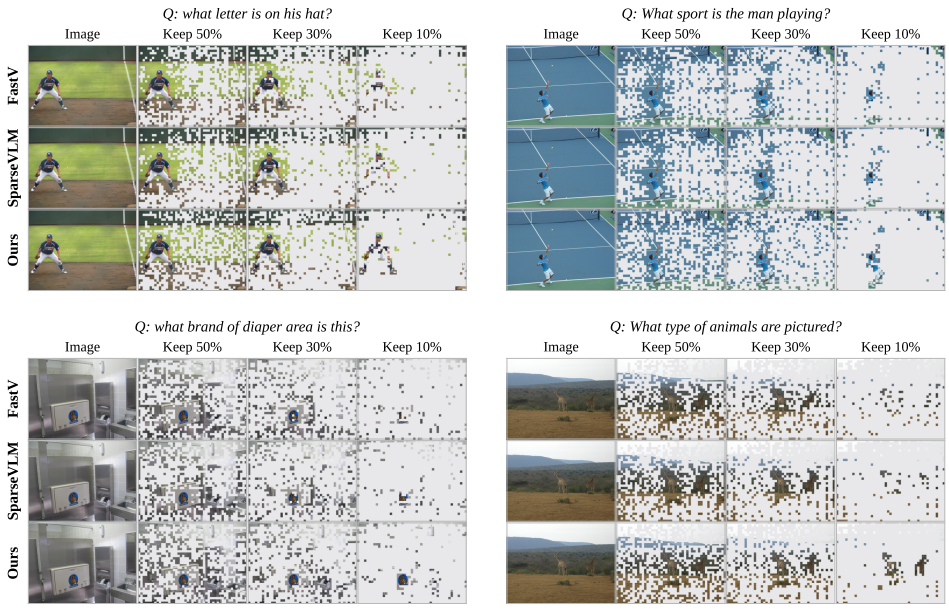


Figure 5: **Additional qualitative samples.** Same protocol as Figure 1: FastV, SparseVLM and SheafPrune at $\rho \in \{0.5, 0.3, 0.1\}$ on LLaVA-NeXT-8B, with pruned positions greyed out.

S LOCAL OPERATOR VISUALIZATIONS

Figure 4 in the main text shows two HR-Bench 8K examples from the existing visualization record, using $t = 5$ and $\delta = 1$. These settings differ from the default $t = 3$ component table. The figure visualizes a local association between degree and corrected score; it does not establish question dependence or a universal failure mechanism.