ALIGN AND BRIDGE: VISUAL TOKEN PRUNING IN THE PRESENCE OF A MODALITY GAP

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models (VLMs) achieve strong performance across multimodal tasks, but their numerous visual tokens incur substantial computational and memory costs. Pruning visual tokens before the LLM decoder reduces the input sequence length, yet identifying query-relevant tokens remains challenging in the presence of a modality gap and limited cross-modal interaction. We propose Align and Bridge, a framework that learns query-conditioned relevance before the decoder and propagates it throughout progressive token pruning. Its nonlinear Vision-Text Aligner (VTA) is trained solely on pre-decoder visual and textual embeddings without executing the LLM decoder. Score-dispersion regularization encourages clearer separation between query-relevant regions and background. Within the decoder, Prior-aware Diverse Relevance (PDR) progressively prunes visual tokens by combining text-to-vision attention with pre-decoder importance priors and token diversity. This design connects early token selection with subsequent pruning to preserve relevant visual information while suppressing redundancy. Experiments across multiple VLMs and multimodal benchmarks show improved performance retention on most evaluated backbones, with substantial reductions in visual tokens and FLOPs; on Qwen2.5-VL, a learned baseline remains stronger. On Qwen3-VL-8B, at a nominal budget of 22.2% of visual tokens averaged across decoder layers, our method preserves 93.4% of the original QA performance and improves relative visual grounding performance by about 12.0 percentage points over the strongest grounding baseline.

1 INTRODUCTION

Vision-language models (VLMs) have made substantial progress across a wide range of multimodal tasks, including visual question answering, reasoning, and knowledge-intensive understanding, by combining visual perception with language understanding (Liu et al., 2023; Bai et al., 2023). Recent VLMs extract large numbers of visual tokens to capture fine-grained visual information, while incorporating mechanisms such as token merging to balance representation quality and computational efficiency (Li et al., 2024; 2023a; Bai et al., 2025a). Nevertheless, hundreds of visual tokens per image are still passed to the large language model (LLM) alongside text tokens, substantially increasing the computational and memory costs of Transformer-based processing.

Visual token pruning addresses this challenge by removing unnecessary visual tokens at different stages of a VLM to reduce computation and accelerate inference. In particular, pre-decoder pruning directly reduces the number of visual tokens entering the LLM, enabling computational savings from the first decoder layer (Zhang et al., 2025b). However, pruning at this stage requires identifying query-relevant visual tokens before sufficient cross-modal interaction has taken place. Although a multimodal projector maps vision encoder outputs into the LLM embedding space, this projection does not necessarily ensure semantic alignment between visual and textual tokens. Even within a shared embedding space, the two modalities can retain distinct feature distributions and imperfect semantic correspondence. Such discrepancies have been observed in contrastively trained multimodal models, including CLIP (Radford et al., 2021) and ConVIRT (Zhang et al., 2022), and have been studied as the modality gap (Liang et al., 2022).

We observe a similar modality gap in VLMs and introduce a Vision-Text Aligner (VTA) to learn query-conditioned relevance in this setting. VTA learns nonlinear alignment corrections from pre-

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.

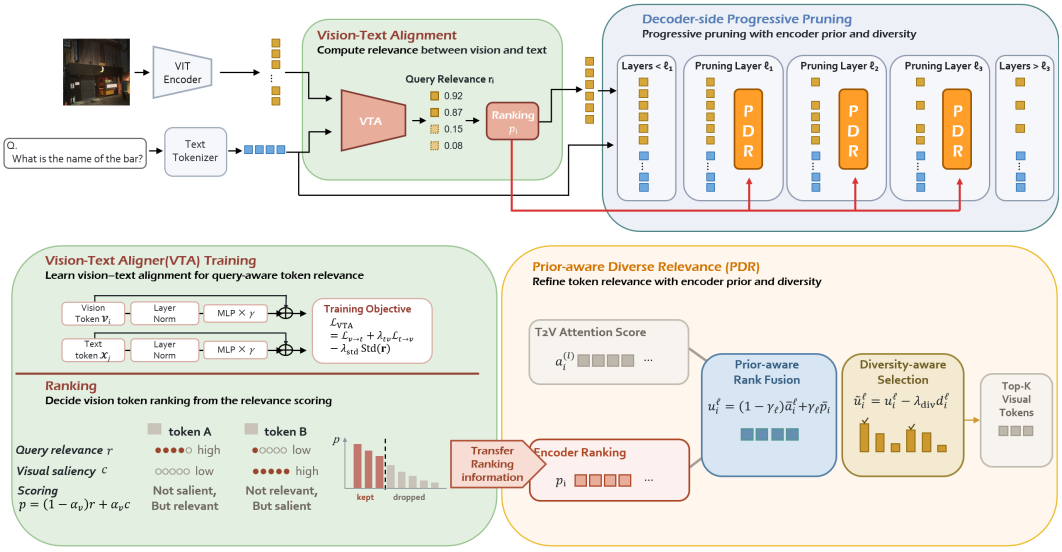


Figure 1: Overview of the VTA+PDR framework. VTA learns query-conditioned relevance from pre-decoder embeddings to select visual tokens. PDR carries the resulting importance scores forward as an encoder prior and combines them with text-to-vision attention and token diversity for progressive pruning.

decoder visual and textual embeddings without executing the LLM decoder during training. A score-dispersion regularizer encourages greater separation between the relevance scores of query-related regions and background. The resulting cross-modal relevance scores guide visual token selection, allowing the model to retain query-relevant visual information and discard unnecessary tokens before they enter the LLM.

Within the decoder, visual tokens interact directly with text tokens, providing contextualized signals of cross-modal relevance. We exploit these interactions through Prior-aware Diverse Relevance (PDR), which progressively reduces the number of visual tokens across decoder layers. At each pruning stage, PDR combines the newly computed text-to-vision relevance with token importance estimated by VTA as a prior. It also accounts for token diversity to discourage the repeated selection of tokens carrying redundant information. This design connects the relevance learned before the decoder with the cross-modal interactions formed within it, preserving query-relevant and complementary visual information throughout progressive pruning. LearnPruner (Takezoe et al., 2026) combines a learned pre-decoder selector with attention-based pruning within the LLM, while MADTP (Cao et al., 2024) learns cross-modal alignment to guide dynamic token pruning across Transformer layers. On the other hand, our approach learns query-conditioned relevance directly from pre-decoder embeddings without involving the LLM decoder during training and explicitly propagates this learned signal to subsequent decoder pruning as a prior.

Our contributions are as follows:

- **VTA:** A nonlinear aligner trained without the LLM decoder improves pre-decoder query-relevant token ranking through score-dispersion regularization.
- **PDR:** VTA-derived importance priors and token diversity guide progressive pruning to preserve relevant visual information across decoder layers.
- We achieve the highest RelAcc. among evaluated methods across the tested token budgets on LLaVA-1.5 and Qwen3-VL-8B. Additional backbone results are provided in the appendix.

2 Related Work

2.1 Vision-Language Models

Modern VLMs commonly project features from pretrained vision encoders into the LLM embedding space. Recent model families, including LLaVA (Liu et al., 2023; 2024a;b; Li et al., 2024), Qwen

(Bai et al., 2023; Wang et al., 2024; Bai et al., 2025b;a), and InternVL (Chen et al., 2024b), support higher input resolutions and richer visual representations. However, encoding complex visual inputs still produces numerous tokens whose processing across LLM layers incurs substantial inference costs, motivating selective visual token reduction.

2.2 Modality Gap

Vision encoders and language models are pretrained with different objectives and data, and projection into a shared embedding space alone does not ensure alignment (Liang et al., 2022). Contrastive objectives have been explored to reduce this *modality gap* by improving cross-modal alignment (Eslami & de Melo, 2025). While these efforts focus on narrowing the gap, we learn query-conditioned relevance from pre-decoder representations to guide visual token pruning despite the gap, without requiring global alignment.

2.3 Visual Token Pruning

Visual token pruning improves VLM efficiency by removing unnecessary visual tokens. Training-free methods use attention, token similarity, or redundancy to prune before the decoder (Zhang et al., 2025b), within it (Xing et al., 2024), or at both stages (Zhang et al., 2025a). Learning-based methods instead train modules to estimate token importance. LearnPruner (Takezoe et al., 2026) combines a learned pre-decoder selector with attention-based pruning inside the LLM, while MADTP (Cao et al., 2024) learns cross-modal alignment to guide dynamic token pruning across Transformer layers. Our approach learns query-conditioned relevance from pre-decoder embeddings using a residual aligner trained without the LLM decoder, and propagates this relevance as a prior for progressive decoder pruning.

3 Method

3.1 Preliminaries and Motivation

Notation. Given an image I and a query Q, visual tokens are

$$V^0 = g(f_v(I)) = \{v_i^0\}_{i=1}^{N_0} \in \mathbb{R}^{N_0 \times D}, \tag{1}$$

where f_v and g denote the vision encoder and projector. Text tokens are $X = \{x_j\}_{j=1}^{T} \in \mathbb{R}^{T \times D}$. The decoder receives $H^{(0)} = [V^0; X]$, and PDR reuses text-to-vision attention at selected layers. The vision encoder, projector, and LLM remain frozen; text tokens are not pruned. VTA retains $K_{\text{enc}}(I, Q)$ visual tokens, which PDR reduces to $K_1 > \cdots > K_S$ across S stages (Figure 1).

Modality gap in VLMs. L2-normalized pre-decoder features from three backbones exhibit near-zero paired cosine similarity (Figure 2(a)); UMAP also separates visual and text representations for LLaVA-1.5 7B and Qwen3-VL 8B (Figure 2(b,c)). This weak absolute alignment does not imply poor relevance ranking, which we test directly in Section 5.2.

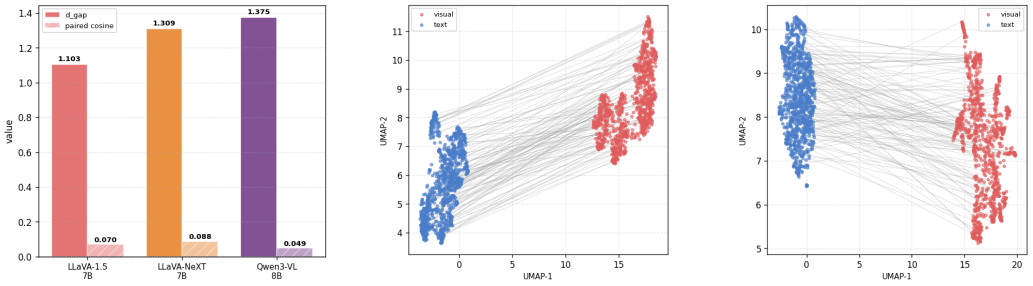
3.2 Vision-Text Aligner

We use a lightweight nonlinear residual aligner to refine weak original similarities while preserving the pretrained representation structure:

$$\tilde{v}_i = v_i^0 + \gamma\, \phi_v\big(\mathrm{LN}(v_i^0)\big), \qquad \tilde{x}_j = x_j + \gamma\, \phi_t\big(\mathrm{LN}(x_j)\big). \tag{2}$$

ϕ_v and ϕ_t are separate three-layer GELU MLPs with dimensions $D \to 512 \to 512 \to D$, where D is the backbone hidden size (4096 or 3584; Table A1), and γ is a learnable scalar distinct from PDR's layer-specific weights γ_ℓ. Text inputs contain only the user query, excluding system prompts, conversation templates, and answer tokens.

Training objective. We cache pre-decoder features and compute a multiple-instance contrastive loss $\mathcal{L}_{v\to t}$ and its reverse $\mathcal{L}_{t\to v}$ using other image–query pairs in the batch as negatives. For image b and



(a) (b) (c)

Figure 2: Modality gap measured from pre-LLM visual and text features of three VLM backbones on 1,000 samples from COCO train2017. (a) Centroid-based d_{gap} and paired cosine similarity across the three backbones. (b) UMAP projection of LLaVA-1.5 7B features. (c) UMAP projection of Qwen3-VL 8B features. In (b) and (c), red and blue denote visual and text tokens, respectively, and gray lines connect features from the same image–caption pair.

candidate query k, let $\text{LME}_{n=1}^{M}(z_n) = \log(M^{-1}\sum_n e^{z_n})$ and define:

$$\sigma_{bik} = \text{LME}_{j=1}^{T_k}\left[\tau^{-1}\cos(\tilde{v}_{bi}, \tilde{x}_{kj})\right], \tag{3}$$

$$S_{bk}^{v\to t} = \log\sum_{i=1}^{N_b}\exp(\sigma_{bik}), \qquad \mathcal{L}_{v\to t} = \frac{1}{B}\sum_{b}\text{CE}\big(S_{b,:}^{v\to t}, b\big)\,. \tag{4}$$

The reverse loss is computed independently, rather than by transposition, so that each query token discriminates its matching image within the batch:

$$\rho_{bjk} = \text{LME}_{i=1}^{N_k}\left[\tau^{-1}\cos(\tilde{v}_{ki}, \tilde{x}_{bj})\right], \tag{5}$$

$$\mathcal{L}_{t\to v} = \frac{1}{B}\sum_{b}\frac{1}{T_b}\sum_{j}\text{CE}(\rho_{bj,:}, b)\,. \tag{6}$$

To discourage uniform relevance scores within each image, we define the overall objective as:

$$\mathcal{L}_{\text{VTA}} = \mathcal{L}_{v\to t} + \lambda_{tv}\,\mathcal{L}_{t\to v} - \lambda_{\text{std}}\,\mathcal{L}_{\text{std}}, \tag{7}$$

where

$$\mathcal{L}_{\text{std}} = \frac{1}{B}\sum_{b=1}^{B}\text{Std}\big(\mathbf{r}^{(b)}\big). \tag{8}$$

Here, λ_{tv} and λ_{std} weight the auxiliary terms. Pooling, cross-entropy, and training details are provided in Appendix A.2.

Query-relevance score. Using the trained aligner, we define each visual token's relevance as its maximum cosine similarity to any query token:

$$r_i = \max_{1\le j\le T}\frac{\tilde{v}_i^{\top}\tilde{x}_j}{\|\tilde{v}_i\|_2\|\tilde{x}_j\|_2}. \tag{9}$$

Max pooling allows different visual regions to match individual query tokens. r_i is used for the dispersion term during training and pre-decoder token selection at inference. **Encoder-side token selection.** We combine relevance r_i and training-free saliency c_i after separately min–max normalizing them within each image. Depending on the backbone, c_i uses token magnitude or CLS→patch attention (Appendix A.3):

$$p_i = (1-\alpha_v)\ \text{Norm}(r_i) + \alpha_v\ \text{Norm}(c_i). \tag{10}$$

Instead of fixed top-k, we derive three budgets from each score distribution: K_{Otsu} counts tokens above the Otsu threshold, K_{gap} stops before the largest descending-score gap, and $K_{\text{top-}p}$ is the smallest count reaching 50% of score mass ($p = 0.5$). The final budget is

$$K_{\text{enc}}(I, Q) = \max\{K_{\text{Otsu}}, K_{\text{gap}}, K_{\text{top-}p}\}\,, \tag{11}$$

yielding a sample-dependent count. We pass the top $K_{\text{enc}}(I, Q)$ tokens and scores p_i to PDR as pre-decoder priors.

3.3 PDR: Prior-aware Diverse Relevance

Decoder text-to-vision importance. At pruning layer ℓ, we average text-to-vision entries of the existing self-attention $A^{(\ell)}$ over H heads and T query tokens:

$$a_i^{(\ell)} = \frac{1}{HT} \sum_{h=1}^{H} \sum_{j=1}^{T} A_{h,j,i}^{(\ell)}. \tag{12}$$

Prior-aware rank fusion. Since encoder scores p_i and decoder scores $a_i^{(\ell)}$ differ in scale, we normalize each using within-image percentile ranks:

$$\bar{a}_i^{\ell} = \text{RankNorm}(a_i^{(\ell)}), \qquad \bar{p}_i = \text{RankNorm}(p_i). \tag{13}$$

Layer-wise importance combines the two ranks:

$$u_i^{\ell} = (1 - \gamma_\ell)\, \bar{a}_i^{\ell} + \gamma_\ell\, \bar{p}_i, \tag{14}$$

where the prior weight γ_ℓ decreases as T2V attention develops in later layers. Rank normalization removes score-scale differences across backbones.

Diversity-aware selection. To prevent top-k selection from favoring redundant patches, we adopt the importance–diversity principle of MMR (Carbonell & Goldstein, 1998) and IDPruner (Tan et al., 2026). We min–max normalize u_i^{ℓ} over candidates and define maximum similarity to the selected set $\mathcal{S}$ as

$$d_i^{\ell} = \max_{j \in \mathcal{S}} \cos(\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)}). \tag{15}$$

The diversity-adjusted score and selection rule are

$$\tilde{u}_i^{\ell} = u_i^{\ell} - \lambda_{\text{div}} d_i^{\ell}, \tag{16}$$

$$i^{\star} = \arg\max_{i \notin \mathcal{S}} \tilde{u}_i^{\ell}, \qquad \mathcal{S} \leftarrow \mathcal{S} \cup \{i^{\star}\}. \tag{17}$$

Here, $\mathbf{h}_i^{(\ell)}$ is the current hidden state. We repeat until the target count is reached, preserving query relevance while reducing spatial and semantic redundancy.

Progressive budgets. To avoid irreversible early errors, PDR refines survivors across decoder layers. Backbone-specific schedules and keep-ratio definitions appear in Appendix A.4–A.6.

4 Experimental Setup

Backbones. We focus on LLaVA-1.5 (7B) and Qwen3-VL (8B); results for LLaVA-NeXT (Liu et al., 2024b), Qwen2-VL, and Qwen2.5-VL appear in the appendix. VTA is trained on a 100k-sample subset of the LLaVA visual instruction tuning dataset (`llava_v1_5_mix665k`; Appendix A.2).

Benchmarks. We evaluate on six benchmark groups: TextVQA (Singh et al., 2019) and GQA (Hudson & Manning, 2019) for VQA, POPE (Li et al., 2023b) for hallucination, ScienceQA (Lu et al., 2022) for scientific reasoning, MME (Fu et al., 2025) for comprehensive evaluation, and RefCOCO/RefCOCO+ (Yu et al., 2016) and RefCOCOg (Mao et al., 2016) for grounding (splits and sample counts in Appendix A.7).

Baselines. We compare with PyramidDrop (Xing et al., 2024), VisPruner (Zhang et al., 2025b), FastV (Chen et al., 2024a), SparseVLM (Zhang et al., 2024), VScan (Zhang et al., 2025a), LearnPruner (Takezoe et al., 2026), and, on Qwen2.5-VL, GlimpsePrune (Zeng et al., 2025).

Settings and efficiency metrics. Backbone-specific token budgets, schedules, and keep-ratio definitions are detailed in Appendix A.5.

Hardware. VTA training and all evaluation and profiling use a single NVIDIA RTX A6000 GPU (46GB); the LearnPruner LPM is trained on four such GPUs (Appendix B.3).

Table 1: LLaVA-1.5 pruning across token budgets. Bold/underline mark the best/second-best value per column. Scores are accuracy (%); MME is the total score and RefCOCO columns are Acc@0.5 (%); relative metrics are normalized to the dense baseline.

Method	TextVQA	POPE	GQA	MME	SQA	RefCOCO	RefCOCO+	RefCOCOg	$\mathrm{rel}_{\mathrm{QA}}$	$\mathrm{rel}_{\mathrm{VG}}$	RelAcc.
Dense baseline, 100% visual tokens											
Vanilla	58.21	85.24	61.98	1867.96	69.51	54.15	46.83	48.93	100.00	100.00	100.00
Nominal budget: 33.3%											
FastV	57.48	83.91	57.85	1733.06	**69.01**	4.07	3.62	3.05	96.52	7.16	63.01
PyramidDrop	56.96	**85.90**	58.53	1783.01	68.42	4.38	3.82	3.25	97.39	7.63	63.73
VisPruner	57.62	85.57	59.41	1793.21	68.47	7.13	6.03	5.67	97.95	12.54	65.92
SparseVLM	**57.73**	85.73	59.20	1751.74	68.77	4.72	4.31	4.12	97.60	8.78	64.29
VScan	56.74	84.04	57.28	1727.20	68.86	2.02	1.93	1.73	96.00	3.80	61.43
LearnPruner	56.35	81.00	60.42	1759.28	68.37	20.73	17.43	17.02	96.37	36.76	74.02
Ours	57.61	84.53	**61.38**	**1796.21**	68.32	**30.72**	**26.84**	**25.72**	**98.32**	**55.54**	**82.28**
Nominal budget: 22.2%											
FastV	56.59	81.09	54.96	1686.88	69.31	1.78	1.90	1.33	94.21	3.35	60.14
PyramidDrop	56.19	83.53	56.65	1743.62	**69.86**	1.84	1.62	1.47	95.95	3.29	61.20
VisPruner	56.83	**85.80**	58.51	1752.33	68.77	3.58	3.29	2.79	97.09	6.45	63.10
SparseVLM	**57.00**	83.03	56.85	1706.99	69.16	1.82	1.44	1.25	95.59	3.00	60.87
VScan	54.99	83.20	55.22	1649.84	68.72	1.18	1.17	1.21	93.67	2.38	59.44
LearnPruner	55.71	80.50	58.94	1726.28	68.12	6.62	5.57	4.94	95.13	11.41	63.73
Ours	56.97	85.08	**60.32**	**1807.23**	68.37	**23.08**	**19.66**	**18.99**	**98.02**	**41.14**	**76.69**
Nominal budget: 11.1%											
FastV	50.58	63.23	46.14	1306.77	**69.91**	0.61	0.26	0.46	81.21	0.87	51.08
PyramidDrop	51.04	77.53	51.96	1435.77	69.01	0.87	0.57	0.72	87.72	1.43	55.36
VisPruner	**55.65**	85.23	56.21	1669.85	68.42	2.54	2.37	2.35	94.82	4.85	61.08
SparseVLM	52.71	70.40	45.07	1420.09	67.92	0.60	0.53	0.50	83.92	1.09	52.86
VScan	49.77	73.49	48.95	1406.76	68.91	0.64	0.42	0.46	85.03	1.01	53.52
LearnPruner	52.65	78.70	55.23	1617.93	68.17	2.17	1.76	1.73	91.31	3.77	58.48
Ours	54.67	**85.59**	**57.80**	**1767.08**	69.71	**5.86**	**5.14**	**4.90**	**96.49**	**10.60**	**64.29**

Table 2: Pruning performance on Qwen3-VL-8B across token budgets. Units follow Table 1.

Method	TextVQA	POPE	GQA	MME	SQA	RefCOCO	RefCOCO+	RefCOCOg	$\mathrm{rel}_{\mathrm{QA}}$	$\mathrm{rel}_{\mathrm{VG}}$	RelAcc.
Dense baseline, 100% visual tokens											
Vanilla	80.21	90.58	62.13	2356.90	94.55	90.51	85.84	87.94	100.00	100.00	100.00
Nominal budget: 33.3%											
FastV	72.86	85.45	56.82	2029.21	41.55	79.87	71.86	76.47	81.33	86.31	83.20
LearnPruner	**78.62**	**88.74**	58.98	**2337.24**	86.07	75.78	69.53	75.19	**96.22**	83.41	91.42
Ours	76.80	88.41	**60.92**	2184.73	**90.93**	**86.94**	**80.77**	**83.77**	96.05	**95.14**	**95.71**
Nominal budget: 22.2%											
FastV	67.78	82.46	53.32	1867.61	40.41	70.62	62.11	67.61	76.67	75.75	76.33
LearnPruner	**74.85**	87.08	56.89	**2311.45**	82.60	65.13	58.80	63.89	93.29	71.04	84.95
Ours	74.44	**88.35**	**59.15**	2110.47	**86.91**	**81.31**	**73.32**	**77.46**	**93.40**	**87.78**	**91.29**
Nominal budget: 11.1%											
FastV	57.26	75.66	48.31	1625.23	40.51	49.33	42.33	45.69	68.89	51.92	62.53
LearnPruner	65.49	81.77	52.07	**2140.72**	78.58	37.65	31.74	32.85	85.93	38.64	68.20
Ours	**66.37**	**87.12**	**56.42**	1962.50	**82.15**	**54.79**	**48.34**	**53.99**	**87.98**	**59.41**	**77.27**

5 RESULTS

5.1 PERFORMANCE ACROSS TOKEN BUDGETS

Tables 1 and 2 report raw and Vanilla-relative scores. RelAcc. averages all eight benchmarks; $\mathrm{rel}_{\mathrm{QA}}$ and $\mathrm{rel}_{\mathrm{VG}}$ average QA and grounding. On LLaVA-1.5, our method achieves the highest RelAcc. at all three token budgets, reaching 82.28% at the 33.3% budget. Here, $\mathrm{rel}_{\mathrm{QA}}$ remains near Vanilla at 98.32%, whereas $\mathrm{rel}_{\mathrm{VG}}$ drops to 55.54%, indicating greater grounding sensitivity to aggressive compression. On Qwen3-VL-8B, our method also leads across the 33.3%–11.1% budgets, with RelAcc. of 95.71%, 91.29%, and 77.27%, respectively. Across these budgets, $\mathrm{rel}_{\mathrm{QA}}$ declines from 96.05% to 87.98%, while $\mathrm{rel}_{\mathrm{VG}}$ falls more sharply from 95.14% to 59.41%, showing stronger grounding sensitivity.

Table 3: Recall@K by scorer on 500 RefCOCOg samples (LLaVA-1.5). VTA uses the main checkpoint (five-epoch schedule, selected at step 200); K is the encoder-stage keep fraction.

K	aligner (VTA)	raw cosine	mean-center
0.1	**0.271**	0.200	0.150
0.2	**0.455**	0.338	0.266
0.3	**0.598**	0.454	0.360
0.5	**0.801**	0.641	0.527
0.7	**0.918**	0.788	0.704

Table 4: RefCOCOg REC accuracy (LLaVA-1.5, encoder-only fixed top-k at keep_ratio=0.20, 5,023 samples; same checkpoint as Table 3).

Method	Acc@0.5 (%)	Mean IoU (%)
aligner (VTA)	**4.82**	**17.61**
raw cosine	4.10	16.50
mean-centering	3.68	15.81

5.2 Validating Relevance Ranking Despite the Modality Gap

Low absolute similarity need not imply poor ranking. On 500 RefCOCOg samples, VTA achieves the highest GT-box Recall@K throughout K=0.1–0.7 (Table 3). We also evaluate all three scorers under 20% encoder-stage pruning on the full 5,023-sample REC test set (Table 4).

Both metrics rank aligner $>$ raw cosine $>$ mean-centering, linking token ranking to downstream grounding. Across four queries on two images (Figure 3), all scorers change their selections with the query, while the aligner produces more query-aligned selections across the four cases; raw cosine, for instance, shifts away from the lady in purple.

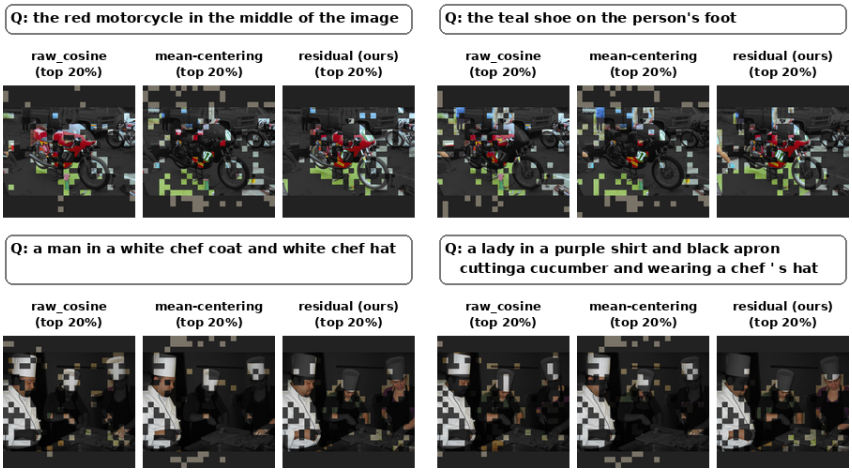


Figure 3: Qualitative comparison of different relevance scorers.

5.3 Ablation Studies

Component ablation. At the nominal 22.2% budget, all variants share VTA+saliency encoder selection and differ only in decoder prior and diversity, so "diversity only" does not remove VTA. As shown in Table 5, combining both components is most effective for the LLaVA backbones, particularly for grounding performance. The trend is backbone-dependent: Qwen2.5-VL favors diversity alone, while Qwen3-VL-8B benefits from the full configuration on RefCOCOg. Complete results appear in Appendix B.4.

Learned scorer at identical token counts. On Qwen3-VL-8B RefCOCOg (nominal 22.2%), with per-sample token counts fixed to the VTA run, replacing VTA with saliency alone or raw cosine yields no statistically significant difference in Acc@0.5 (77.46 vs. 77.56/77.86); VTA's gain appears in mean IoU (+1.40/+0.80). This isolates the scorer within the VTA-derived budget, not VTA's adaptive budgeting (Appendix B.7).

VTA training ablation. We further ablate VTA's objective and projection on LLaVA-1.5. Although variants train for three or five epochs, the same held-out criterion selects every checkpoint

Table 5: Component ablation at the nominal 22.2% budget with the full pipeline, main checkpoints, and main-table schedules. Encoder selection is VTA+saliency in all variants. Values are RefCOCOg Acc@0.5.

Model	Div. only	Enc. prior	Full (Ours)
LLaVA-1.5	15.83	16.32	**18.99**
LLaVA-NeXT	29.68	40.14	**47.32**
Qwen2.5-VL	**64.78**	56.42	58.49
Qwen3-VL-8B	74.58	76.59	**77.46**

Table 6: VTA training ablation on LLaVA-1.5. Recall@0.2 is GT-box recall at encoder keep fraction $K{=}0.2$ (500 samples); TextVQA and MME use encoder-only fixed top-k at keep_ratio=0.5. All checkpoints are selected by the same criterion within the first epoch (Appendix B.8).

Variant	Recall@0.2	TextVQA	MME
Ours	**0.455**	56.53	1775.73
No residual	0.451	56.60	1795.23
No reverse loss	0.442	**56.95**	**1811.75**
No dispersion	0.431	54.40	1766.40
Linear projection	0.124	53.82	1643.67

within epoch one and does not improve later. As shown in Table 6, removing the dispersion loss lowers Recall@K at every K and REC accuracy at both budgets. A linear projection sharply lowers Recall@K, TextVQA, and MME. The nonlinear projection's downstream advantage varies with token budget: at keep_ratio=0.5 the linear variant reaches 20.98 REC Acc@0.5, above the residual model's 20.55. The independent benefit of the residual connection also depends on the budget and metric, and removing the reverse loss does not consistently reduce performance. Thus, dispersion is the most consistent component. Full Recall@K and downstream results appear in Appendix B.8.

5.4 Inference Efficiency

At the nominal 11.1% budget, our method is more accurate than LearnPruner on both backbones (Table 7). Measured retention is 11.4% and 11.6% for our method versus 11.2% and 18.4% for LearnPruner on LLaVA-1.5 and Qwen3-VL-8B. Against dense inference, it speeds up prefill on both and end-to-end LLaVA-1.5 by 1.12×, but trails LearnPruner. On Qwen3-VL-8B, pruning reduces FLOPs to 0.2606 and accelerates prefill by 1.28×, whereas decoding runs at 0.92×; combined, this yields a 0.95× end-to-end speedup. Increased decoding time thus outweighs the prefill gain, exposing an accuracy–latency trade-off. Full profiling appears in Appendix B.3.

Table 7: Accuracy-efficiency comparison on RefCOCOg at the nominal 11.1% budget. Kept is the measured layer-weighted visual-token retention ρ relative to dense inference (Appendix A.5), counting each cut where it takes effect in the respective implementation. FLOPs and KV cache account for per-layer token counts. Full comparisons appear in Appendix B.3.

Backbone	Method	RefCOCOg	Kept (%)	FLOPs	Speedup	Prefill	Peak (MB)	KV (MB)
LLaVA-1.5	No pruning (dense)	48.93	100.0	1.0000	1.00×	1.00×	15617.8	288.0
	LearnPruner (LPM)	1.73	11.2	**0.2404**	**1.17**×	**2.45**×	15016.4	**32.4**
	Ours (VTA+PDR)	**4.90**	11.4	0.2420	1.12×	2.09×	**14348.0**	32.8
Qwen3-VL-8B	No pruning (dense)	87.94	100.0	1.0000	1.00×	1.00×	17646.4	38.2
	LearnPruner (LPM)	32.85	18.4	0.3123	**1.04**×	**1.95**×	**17606.0**	7.0
	Ours (VTA+PDR)	**53.99**	11.6	**0.2606**	0.95×	1.28×	17637.0	**4.4**

5.5 Qualitative Analysis

Changing only the query redirects selection to the standing man, rider, or right motorcycle (Figure 4). IoUs of 0.83, 0.69, and 0.61 indicate referent-specific salience; further cases appear in Appendix B.9.

6 Discussion and Limitations

Interpreting the results. At 22.2% on Qwen3-VL-8B, $\mathrm{rel}_{\mathrm{QA}}$ remains 93.40%, while $\mathrm{rel}_{\mathrm{VG}}$ exceeds the strongest baseline by 12 points, showing stronger local grounding under compression. Recall@K and query-dependent selections support learned relevance. At identical token counts, VTA improves mean IoU but not Acc@0.5 over simpler scorers, so its effect is smaller than the full-method gain (Appendix B.7).

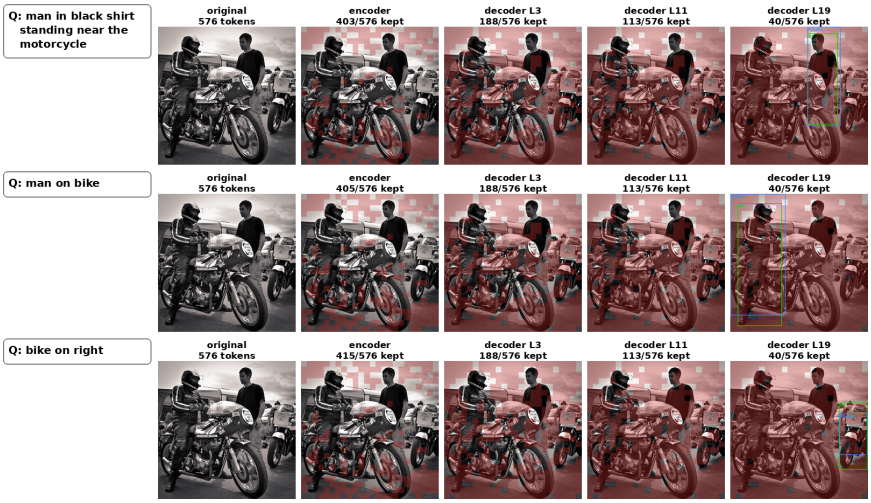


Figure 4: Query-dependent selection on the same image (LLaVA-1.5, RefCOCO testA, the reported 22.2% setting). Pruned tokens are tinted red. The encoder output count varies by query, and the decoder keeps a fixed 188/113/40 tokens at layers 3/11/19. Surviving tokens follow each query's referent (the standing man, the rider, and the motorcycle on the right), and the predicted (blue) and ground-truth (green) boxes overlap with IoU ≥ 0.5 for all three queries.

Roles of the two stages. PDR combines VTA's pre-decoder relevance with T2V attention and diversity to preserve evidence while reducing redundancy. Removing dispersion lowers Recall@K and REC accuracy at both budgets despite slight SQA and POPE gains; reverse-loss and residual effects vary by metric, making dispersion the most consistent component.

Limitations and future work. Fixed family-specific schedules cannot adapt to visual or query difficulty. VTA alignment and PDR diversity can offset token savings, while α_v, γ_ℓ, λ_{div}, and schedules require per-backbone tuning (Table 7). Adaptive schedules, automatic calibration, lighter selectors, and fused attention/MMR could improve both accuracy and latency.

7 Conclusion

We propose VTA, which learns query-conditioned relevance from pre-LLM embeddings without executing the decoder, and PDR, which propagates it as a prior for decoder-stage pruning. Nonlinear alignment with score dispersion improves VTA ranking, while PDR combines the prior with T2V attention and MMR diversity. This shared signal connects pre-decoder selection and decoder-stage pruning, allowing early query relevance to guide token retention while later layers refine the surviving evidence.

VTA achieves higher GT-box Recall@K than raw cosine and mean-centering on LLaVA-1.5, while changing the query shifts surviving tokens toward the corresponding target. Ablations identify score dispersion as the most consistent contributor to ranking and REC accuracy, whereas the effects of reverse loss and residual alignment vary with metric and budget. Our method also achieves the highest RelAcc. among evaluated methods at all tested budgets on LLaVA-1.5 and Qwen3-VL-8B. At a nominal layer-averaged budget of 22.2% of visual tokens, Qwen3-VL-8B achieves rel_{QA} of 93.40%, rel_{VG} of 87.78%, and RelAcc. of 91.29%. These results highlight reduced loss of query-specific visual evidence under aggressive compression, rather than uniform gains across QA tasks, and show that grounding remains more sensitive than QA as the budget shrinks.

Our method accelerates dense prefill and end-to-end inference on some backbones but trails lighter selectors, reflecting an accuracy–speed trade-off. Profiling shows that fewer visual tokens reduce prefill FLOPs and KV-cache use, while selection overhead and decoding can limit end-to-end gains. Adaptive budgets, automatic calibration, lighter selectors, and fused selection kernels could reduce this overhead while preserving relevance.

AI USE STATEMENT

AI language models assisted with text editing, some figures, and experimental code implementation. The authors designed, conducted, and analyzed all experiments and made all methodological and experimental decisions. All AI-generated or assisted text, figures, and code were reviewed for accuracy and consistency with the paper.

ETHICS STATEMENT

We use only public VLM benchmarks and pretrained checkpoints (Section 4), collecting no new human-subject data or personally identifiable information. We follow the datasets' and models' licenses and terms of use, and refer to the original dataset documentation for inherent biases and labeling practices. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

VTA's aligner architecture and objectives appear in Section 3.2, with training hyperparameters in Appendix A.2 (Table A1). PDR's prior fusion and diversity-aware selection are described in Section 3.3, and encoder-side adaptive selection in Appendix A.3. Decoder schedules are provided for LLaVA in Appendix A.4–A.5 (Tables A4 and A6; fixed token counts for LLaVA-1.5, ratio-mode for LLaVA-NeXT), and for Qwen in Appendix A.6 (Table A7). Backbones, benchmarks, baselines, and keep-ratio tiers are specified in Section 4. Implementation and experimental settings are documented in these sections as fully as anonymity permits.

REFERENCES

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

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

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

Jianjian Cao, Peng Ye, Shengze Li, Chong Yu, Yansong Tang, Jiwen Lu, and Tao Chen. Madtp: Multimodal alignment-guided dynamic token pruning for accelerating vision-language transformer. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15710–15719. IEEE, 2024.

Jaime G Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pp. 335–336, 1998.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024a.

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

Sedigheh Eslami and Gerard de Melo. Mitigate the gap: Improving cross-modal alignment in CLIP. In *International Conference on Learning Representations*, 2025.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *Advances in Neural Information Processing Systems*, 38, 2025.

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

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023a.

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

Victor 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. *Advances in Neural Information Processing Systems*, 35:17612–17625, 2022.

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

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26286–26296. IEEE, 2024a.

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

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

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

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

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

Rinyoichi Takezoe, Yaqian Li, Zi-Hao Bo, Anzhou Hou, Mo Guang, and Kaiwen Long. Learn-pruner: Rethinking attention-based token pruning in vision language models. In *International Conference on Learning Representations*, volume 2026, pp. 66381–66400, 2026.

Yifan Tan, Yifu Sun, Shirui Huang, Hong Liu, Guanghua Yu, Jianchen Zhu, and Yangdong Deng. Idpruner: Harmonizing importance and diversity in visual token pruning for mllms. *arXiv preprint arXiv:2602.13315*, 2026.

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

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

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European conference on computer vision*, pp. 69–85. Springer, 2016.

Quan-Sheng Zeng, Yunheng Li, Qilong Wang, Peng-Tao Jiang, Zuxuan Wu, and Ming-Ming Cheng. A glimpse to compress: Dynamic visual token pruning for large vision-language models. *arXiv preprint arXiv:2508.01548*, 2025.

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

Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20857–20867. IEEE, 2025b.

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

Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D Manning, and Curtis P Langlotz. Contrastive learning of medical visual representations from paired images and text. In *Machine learning for healthcare conference*, pp. 2–25. PMLR, 2022.

A Supplementary Experimental Details

This appendix provides the numerical details omitted from Section 3. It covers aligner training in Section A.2, encoder-side selection by backbone in Section A.3, the three-stage LLaVA schedules in Section A.4 and A.5, and the five-stage Qwen schedule in Section A.6.

A.1 Complexity

The aligner scores only pre-LLM visual and query embeddings. Its cost is linear in the number of visual and text tokens except for the $O(N_0T)$ similarity matrix, and is small relative to processing the full multimodal sequence through every LLM layer. Decoder pruning reuses the existing attention projections for text-to-vision relevance. Diversity selection adds a visual-token similarity matrix only over the shrinking candidate set. This overhead is partly offset by lower costs in later layers. We report measured prefill latency, end-to-end speedup, memory, and layer-weighted FLOPs.

A.2 Aligner Training Details

Equations 4 and 6 define batchwise pooling and cross-entropy computation.

The main checkpoints and the plain variant use a five-epoch schedule, and the lam0, std0, and linear variants use a three-epoch schedule. The held-out matched-versus-mismatched gap peaks within the first epoch for every backbone and variant and is not improved afterwards. Since the training trend saturates early, all compared checkpoints come from early in training. Because the learning rate follows a cosine schedule over the configured epochs, the two settings still differ slightly at the same step (by about 1.5% in learning rate at step 200 on LLaVA-1.5), so the selected weights are not identical across epoch settings. Table A2 summarizes the checkpoint, encoder selection, keep ratio, and PDR usage of each experiment group. All VTA checkpoints are trained on the same 100k subset.

Table A1: Aligner training hyperparameters. Values are shared across backbones unless noted otherwise.

Setting	Value	Setting	Value
Backbone parameters	Frozen	Optimizer	AdamW
Activation	GELU	Learning rate	1e-4
Initialization	He normal	Epochs	5 (lam0/std0/linear: 3)
Residual scale γ	Learnable	Contrastive temperature τ	0.07
Reverse-direction weight λ_{tv}	0.5	Score-dispersion weight λ_{std}	0.3
Batch size	128 (LLaVA), 32 (Qwen)	Checkpoint evaluation interval	100 steps

Aligner architecture: residual dual MLP (ϕ_v, ϕ_t), $D \to 512 \to 512 \to D$; $D = 4096$ for LLaVA-1.5, LLaVA-NeXT, and Qwen3-VL-8B (8.94M parameters), $D = 3584$ for Qwen2-VL and Qwen2.5-VL (7.89M parameters)
Backbone checkpoints: `liuhaotian/llava-v1.5-7b`, `liuhaotian/llava-v1.6-vicuna-7b`, `Qwen/Qwen2-VL-7B-Instruct`, `Qwen/Qwen2.5-VL-7B-Instruct`, `Qwen/Qwen3-VL-8B-Instruct`
Text input: user query only, excluding system and template tokens
Training data: 100k subset of `llava_v1_5_mix665k` sampled with seed 42 by source (COCO 45%, VG 20%, OCR-VQA 15%, GQA 15%, TextVQA 5%)
Checkpoint selection: maximum matched-versus-mismatched query relevance gap on a held-out split of the same subset (500 pairs; 2,000 for Qwen2/2.5-VL). Evaluation-benchmark labels and performance are not used for selection or tuning; per-benchmark budget calibration uses only the benchmarks' visual-token counts (dense and encoder output) to match the nominal budget.
Selected checkpoints all fall within the first epoch: step 200 (LLaVA-1.5, LLaVA-NeXT), 400 (Qwen2/2.5-VL), 900 (Qwen3-VL-8B).

Table A2: Settings of each experiment group. "Main ckpt" denotes the per-backbone checkpoint in Table A1 (five-epoch schedule, selected within the first epoch).

Experiment group	Checkpoint	Encoder selection	Keep ratio	PDR
Main results, Tables 1, 2, A9–A11	Main ckpt	Adaptive maxkeep, VTA + saliency (α_v=0.5)	33.3–11.1% tiers	Yes
Component ablation, Tables 5, A14	Main ckpt	Same as main	22.2%	Prior/diversity toggled
Efficiency, Table 7	Main ckpt	Same as main	11.1%	Yes
Recall@K and REC, Tables 3, 4	LLaVA-1.5 main ckpt	Scorer only, fixed top-k	K=0.1–0.7; REC 0.2	No
VTA vs. saliency, Table A16	Main ckpt	Adaptive maxkeep, $\alpha_v \in \{0, 1\}$	0.5 (lower bound)	No
Scorer control, Table A17	Qwen3-VL-8B main ckpt	Scorer swapped, per-sample counts fixed to Ours	22.2%	Yes
VTA training ablation, Tables 6, A18–A21	LLaVA-1.5 variants (3- or 5-epoch schedule)	Aligner only, fixed top-k	0.2, 0.5	No

A.3 Encoder-Side Selection by Backbone

The encoder score in Eq. 10, $p_i = (1 - \alpha_v)\,\mathrm{Norm}(r_i) + \alpha_v\,\mathrm{Norm}(c_i)$, combines the learned aligner score r_i with a backbone-specific saliency signal c_i. The adopted settings are listed below.

Both LLaVA backbones use CLS-to-patch attention for c_i, while all three Qwen backbones use the visual-token L_2 norm. The value 0.5 is the cumulative score-mass threshold for top-p, not a fixed keep ratio. For each sample, we compute the token counts proposed by Otsu, the largest gap in descending scores, and top-$p = 0.5$, then use their maximum. Encoder input and output counts can therefore vary by sample. The original count N_0 also varies for variable-resolution backbones. The numerator and denominator in the table are measured dataset means, so their ratio need not equal 0.5.

A.4 LLaVA Decoder-Side Configuration

This section and Section A.5 apply only to LLaVA-1.5 and LLaVA-NeXT, which use three pruning layers instead of the five used for the Qwen family (Section A.6).

The two LLaVA backbones use different schedule types. LLaVA-1.5 uses a fixed target count per stage because every image yields the same $N_0 = 576$ tokens, so a fixed count is a fixed fraction of the dense count. Each stage keeps $\min(\text{target}, n)$ tokens, where n is the current number of surviving visual tokens; if n does not exceed the target, that stage removes nothing. Adaptive encoder selection keeps at least $0.5N_0 = 288$ tokens (e.g., 288–449 tokens on TextVQA at the 22.2% tier), so at the 22.2% and 11.1% tiers, whose targets are below 288, every sample reaches the target counts. At the 33.3% tier, the first target of 309 can exceed the encoder output: across the ten benchmark splits of the main setting (43,460 samples), 47.9% of samples keep all of their tokens at layer 3, while the later targets (206 and 71) apply to every sample. Backbones with a variable N_0 (LLaVA-NeXT and the Qwen family) use the ratio-mode schedule instead, since a fixed count would realize different retention rates on benchmarks with different image resolutions: each stage keeps a fixed proportion of the tokens that are live for that sample.

Table A3: Encoder-side settings and measured mean token counts after adaptive selection. The selector uses the largest count proposed by Otsu, max-gap, and top-$p = 0.5$. Counts are measured from per-sample logs of the reported configuration on TextVQA val (5,000 samples): mean encoder output / mean input count, with the ratio in parentheses. For variable-resolution backbones, the per-sample ratio averaged over samples differs from this ratio of means by at most about one percentage point.

Backbone	`select_mode`	Saliency signal c_i	α_v	Mean kept/input
LLaVA-1.5	`cls_fusion`	ViT CLS→patch attention	0.50	≈304 / 576 (52.8%)
LLaVA-NeXT	`cls_fusion`	ViT CLS→patch attention	0.50	≈1156 / 2310 (50.0%)
Qwen2-VL	`aligner`	Visual-token L_2 norm	0.50	≈660 / 985 (67.0%)
Qwen2.5-VL	`aligner`	Visual-token L_2 norm	0.50	≈522 / 985 (53.0%)
Qwen3-VL-8B	`aligner`	Visual-token L_2 norm	0.50	≈420 / 750 (56.0%)

Table A4: Shared decoder-side pruning settings for both LLaVA backbones and all reported tiers.

Setting	Value
Pruning layers $\mathcal{P}$	$\{3, 11, 19\}$
Encoder-prior schedule $(\gamma_3, \gamma_{11}, \gamma_{19})$	0.60, 0.35, 0.20
Diversity coefficient λ_{div}	0.5

A.5 LLaVA Decoder Token Schedules by Backbone and Tier

Keep-ratio definition (all backbones). For sample j with $N_0^{(j)}$ dense visual tokens, let $n_0^{(j)}$ be the encoder-stage output and $n_s^{(j)}$ the count after the s-th decoder pruning layer. With L decoder layers and w_s the number of layers that process $n_s^{(j)}$ tokens,

$$\rho^{(j)} = \frac{\sum_s w_s \, n_s^{(j)}}{L \, N_0^{(j)}}, \qquad \rho = \frac{1}{M} \sum_{j=1}^{M} \rho^{(j)}.$$

Layers are indexed from 0. On every backbone, the decoder score at layer ℓ comes from layer ℓ's own attention: layer ℓ's query and key projections applied to its input hidden states. Only the removal point differs. LLaVA-1.5 runs layer ℓ on all tokens and then removes tokens, so the cut takes effect after layer ℓ. The Qwen family computes the same score before running layer ℓ, removes tokens, and then runs layer ℓ on the survivors. LLaVA-NeXT runs layer ℓ once on all tokens to obtain its attention, removes tokens, and reruns layer ℓ on the survivors, so the cut takes effect before layer ℓ. FLOPs and ρ use the same removal points; like the selector, the extra scoring pass on LLaVA-NeXT is not counted in FLOPs. The tier labels (33.3%, 22.2%, and 11.1%) are nominal budgets. Per-benchmark measured ρ at the 33.3/22.2/11.1% tiers ranges over 32.6–33.5/22.1–22.5/11.1–11.5% for LLaVA-1.5 (five QA benchmarks), 30.2–31.9/22.8–24.4/11.5–12.6% for LLaVA-NeXT (ten benchmarks), 31.4–37.0/22.2–22.3/about 11.1% for Qwen2-VL, 29.3–40.6/22.1–22.3/11.1–11.2% for Qwen2.5-VL, and 31.8–36.5/21.2–24.4/10.6–12.1% for Qwen3-VL-8B (ten benchmarks for the Qwen family). Qwen2-VL and Qwen2.5-VL calibrate r_1 per benchmark at the 22.2% tier and below, so ρ closely matches the nominal budget there. Qwen3-VL-8B, and the 33.3% tier on most benchmarks, share one r_1 across benchmarks, so ρ varies across benchmarks. Baseline schedules are set per benchmark to the nominal budget using that benchmark's mean dense token count, so their per-sample measured retention can differ from it. In particular, LearnPruner on Qwen3-VL-8B sets r_1/r_2 by the final-stage ratio r_2/N_0, so its layer-weighted measured retention exceeds the nominal budget: 18.4% on RefCOCOg at the 11.1% tier (Table 7) and 36.8–37.0% on the grounding benchmarks at the 22.2% tier (Table A5). The main comparisons are therefore nominal-budget comparisons, not comparisons at identical token counts or identical compute. Because ρ averages over all decoder layers, it exceeds the fraction kept at the final stage. Since the decoder only removes tokens, ρ never exceeds the encoder-stage ratio in Table A3.

Table A5: Measured layer-weighted retention ρ (%, mean of per-sample ratios over each full split) on the grounding benchmarks for Qwen3-VL-8B at the nominal 22.2% budget, computed from the per-sample token counts logged by the existing evaluation runs. FastV processes all tokens in layers 0–1; LearnPruner processes $\min(r_1, N_0)$ tokens in layers 0–11 and $\min(r_2, r_1)$ in layers 12–35.

Method	RefCOCO testA	RefCOCO testB	RefCOCO+ testA	RefCOCO+ testB	RefCOCOg test
FastV	26.6	26.6	26.6	26.6	26.6
LearnPruner	37.0	36.9	37.0	36.9	36.8
Ours	23.7	23.7	23.8	23.6	23.2

For LLaVA-1.5 we list the fixed token count at each layer and its fraction of $N_0 = 576$. For LLaVA-NeXT the ratios are relative to the preceding stage; the first ratio is relative to the tokens that survive encoder-side selection.

Table A6: Decoder-stage schedules for each LLaVA backbone and tier. LLaVA-1.5: target token count after each pruning layer (fraction of $N_0 = 576$ in parentheses); the realized count is $\min(\text{target}, n)$ for n surviving tokens, and only the first 33.3% target (309) can exceed the encoder output. LLaVA-NeXT (ratio-mode): keep ratio relative to the preceding stage. The backbone labels include the number of decoder layers.

Backbone	Keep ratio	ℓ_1=3	ℓ_2=11	ℓ_3=19
LLaVA-1.5 (32 L)	33.3%	309 (53.6%)	206 (35.8%)	71 (12.3%)
	22.2%	188 (32.6%)	113 (19.6%)	40 (6.9%)
	11.1%	51 (8.9%)	34 (5.9%)	15 (2.6%)
LLaVA-NeXT (32 L)	33.3%	93.8%	86.6%	39.9%
	22.2%	77.9%	60.1%	40.3%
	11.1%	27.0%	67.3%	46.6%

A.6 Qwen Decoder-Side Configuration: Ratio-Mode Schedule

For Qwen2-VL, Qwen2.5-VL, and Qwen3-VL-8B, each stage retains a fixed proportion of the current live tokens for that sample. The first-stage ratio r_1 is relative to the tokens that survive encoder-side selection.

Table A7: Five-stage ratio-mode decoder settings for the Qwen family.

Setting	Value
Qwen2/2.5-VL pruning layers, 28 decoder layers	$\{1, 5, 9, 15, 23\}$
Qwen3-VL-8B pruning layers, 36 decoder layers	$\{1, 6, 12, 19, 30\}$
Stage 2–5 relative keep-ratio chain	$(0.880, 0.818, 0.722, 0.538)$
Encoder saliency weight α_v	0.50
5-stage encoder-prior schedule	$(0.60, 0.48, 0.35, 0.27, 0.20)$
Diversity coefficient λ_{div}	0.5

A.7 Implementation Details

GT-box Recall@K. We map each ground-truth box onto the 24×24 patch grid after the same square padding used by LLaVA preprocessing, and treat every patch that overlaps the box as a ground-truth token. Per-sample recall is $|\text{top-}K \cap \text{GT}|/|\text{GT}|$, averaged over samples. The 500 analysis samples are the first 500 RefCOCOg test samples after a random permutation with seed 0. For Qwen2-VL in Table A15, images are not padded; the box is scaled directly onto the merged-token grid with the same overlap rule, and the grid shape is taken as the factor pair of the visual-token count whose aspect ratio is closest to that of the image.

Table A8: Final stage-1 relative keep ratio r_1 for each Qwen ratio-mode tier. Qwen2-VL and Qwen2.5-VL calibrate r_1 per benchmark at the 22.2% tier and below, so the range over the ten benchmarks is listed. At the 33.3% tier, Qwen2-VL uses a single value and Qwen2.5-VL uses 0.956 on most benchmarks (0.870 on RefCOCOg and 0.797 on RefCOCO+ testB). Qwen3-VL-8B uses a single value for all benchmarks.

Nominal keep ratio	r_1 (Qwen2-VL)	r_1 (Qwen2.5-VL)	r_1 (Qwen3-VL-8B)
33.3%	0.752	0.797–0.956	0.859
22.2%	0.428–0.514	0.496–0.631	0.558
11.1%	0.186–0.229	0.219–0.287	0.257

Mean-centering. The mean-centering baseline in Tables 3 and 4 and Figure 3 is training-free. A per-dimension global mean of all visual tokens and of all text tokens is computed once for each modality over 1,000 `llava_v1_5_mix665k` (image, query) pairs and kept fixed; at inference, each modality's mean is subtracted and the cosine score is computed as for raw cosine (max over query tokens). It has no learned parameters, so the comparison is between the learned aligner and training-free global mean removal.

Decoder input. The decoder receives the original projected visual embeddings of the selected tokens, not the aligner outputs; the aligner is used only for scoring.

Normalization and MMR. Norm in the encoder score fusion is per-image min-max normalization. PDR's RankNorm recomputes ranks over the currently surviving tokens at every pruning stage and maps them to $[0, 1]$; the encoder prior is sliced to the survivors accordingly, and ties follow the sort order. MMR operates on min-max-normalized scores and, with an empty selected set, first picks the highest-scoring token.

Position information. For the Qwen family, surviving tokens keep their original M-RoPE position IDs at both stages, and Qwen3-VL-8B's DeepStack visual features are sliced with the same keep mask before injection. For LLaVA, the sequence after encoder-side selection uses contiguous position IDs, which are then preserved through decoder-stage pruning.

Efficiency metrics. The FLOPs ratio counts theoretical decoder FLOPs using per-layer token counts and excludes the vision encoder and the selector. KV cache is the layer-weighted size for visual tokens only.

Evaluation. All methods use the same question files and images and follow the official evaluation protocol of each benchmark. For the RefCOCO family, Acc@0.5 uses the same box IoU as the official REFER evaluation. The RefCOCO and RefCOCO+ columns are the unweighted mean of testA and testB (1,975 and 1,810 samples for RefCOCO; 1,975 and 1,798 for RefCOCO+), and RefCOCOg uses its test split (5,023 samples). The QA benchmarks use TextVQA validation (5,000), POPE on COCO val2014 with its adversarial, popular, and random subsets (8,910), GQA test-dev balanced (12,578), MME (2,374), and ScienceQA-IMG test (2,017).

B Additional Results

We report detailed LLaVA-NeXT results in Section B.1 and backbone-specific results for Qwen2-VL and Qwen2.5-VL in Section B.2. These tables were omitted from the main paper due to space constraints.

B.1 Detailed LLaVA-NeXT Results

Table A9 shows that our method achieves the highest RelAcc. at all three nominal budgets, reaching 88.81%, 85.55%, and 71.98% at the nominal 33.3%, 22.2%, and 11.1% budgets, respectively. At the nominal 22.2% budget, Ours exceeds LearnPruner by about 11.8 percentage points in RelAcc. (85.55% vs. 73.76%). Its $\mathrm{rel}_{\mathrm{QA}}$ remains close to the dense baseline at 98.96%, while $\mathrm{rel}_{\mathrm{VG}}$ reaches 63.20%, outperforming LearnPruner by 24.65 percentage points. Even at the nominal 11.1% budget, Ours preserves 95.30% $\mathrm{rel}_{\mathrm{QA}}$ and achieves the highest $\mathrm{rel}_{\mathrm{VG}}$ among the compared methods at

Table A9: Visual-token pruning performance on LLaVA-NeXT. Units follow Table 1.

Method	TextVQA	POPE	GQA	MME	SQA	RefCOCO	RefCOCO+	RefCOCOg	$\mathrm{rel}_{\mathrm{QA}}$	$\mathrm{rel}_{\mathrm{VG}}$	RelAcc.
Dense baseline, 100% visual tokens											
Vanilla	61.37	88.76	64.42	1851.19	70.15	83.49	75.25	80.41	100.00	100.00	100.00
Nominal budget: 33.3%											
PyramidDrop	60.39	**89.07**	63.21	1804.20	68.57	40.34	34.47	36.27	98.42	46.41	78.91
VisPruner	54.07	87.30	62.82	1783.22	**69.61**	2.41	2.30	3.26	95.91	3.33	61.19
VScan	57.79	87.90	61.73	1768.58	67.92	22.84	18.05	17.80	96.28	24.49	69.36
LearnPruner	57.81	88.70	63.75	1799.85	67.97	57.63	49.66	50.17	97.44	65.80	85.58
Ours	**60.52**	89.01	**64.39**	**1831.07**	68.82	**61.80**	**54.33**	**55.01**	**99.17**	**71.54**	**88.81**
Nominal budget: 22.2%											
PyramidDrop	59.12	88.20	61.35	1803.04	67.92	22.92	19.15	26.68	97.03	28.69	71.40
VisPruner	54.12	87.43	62.47	1786.12	**69.41**	2.60	2.54	3.46	95.82	3.60	61.24
VScan	56.42	85.73	59.34	1742.79	67.18	16.55	13.51	13.02	94.11	17.99	65.56
LearnPruner	55.51	87.58	61.19	1726.47	68.07	34.28	29.02	28.97	94.88	38.55	73.76
Ours	**60.38**	**89.16**	**64.03**	**1828.39**	68.62	**55.73**	**48.17**	**47.32**	**98.96**	**63.20**	**85.55**
Nominal budget: 11.1%											
PyramidDrop	53.86	80.97	56.06	1669.41	67.53	9.78	7.28	8.18	90.49	10.52	60.50
VisPruner	54.17	86.10	**61.32**	**1790.44**	**69.21**	2.62	2.25	3.46	95.17	3.48	60.78
VScan	50.92	71.93	53.12	1481.55	67.82	8.05	6.69	6.93	84.64	9.05	56.29
LearnPruner	48.15	80.05	55.57	1562.44	67.13	10.50	8.93	9.85	87.00	12.23	58.96
Ours	**55.97**	**88.61**	61.00	1738.07	67.97	**30.16**	**24.82**	**24.29**	**95.30**	**33.11**	**71.98**

33.11%. These results show that Ours maintains general QA performance while better preserving visual evidence required for grounding under aggressive pruning.

B.2 Detailed Qwen2-VL and Qwen2.5-VL Results

Table A10: Visual-token pruning performance on Qwen2-VL. Units follow Table 1.

Method	TextVQA	POPE	GQA	MME	SQA	RefCOCO	RefCOCO+	RefCOCOg	$\mathrm{rel}_{\mathrm{QA}}$	$\mathrm{rel}_{\mathrm{VG}}$	RelAcc.
Dense baseline, 100% visual tokens											
Vanilla	82.11	89.76	62.33	2340.78	85.72	76.64	71.75	73.22	100.00	100.00	100.00
Nominal budget: 33.3%											
FastV	**80.61**	85.33	58.33	**2194.15**	47.25	31.21	28.09	30.04	87.14	40.30	69.57
Ours	74.03	**87.22**	**60.05**	2070.28	**80.47**	**52.34**	**48.13**	**47.72**	**93.20**	**66.85**	**83.32**
Nominal budget: 22.2%											
FastV	77.82	85.53	60.26	**2314.65**	44.52	12.19	10.56	11.73	87.51	15.55	60.53
SparseVLM	**78.55**	85.33	**61.10**	2078.79	47.25	13.98	10.94	13.96	86.54	17.52	60.65
Ours	70.93	**86.31**	58.10	1987.62	**79.47**	**35.21**	**31.66**	**33.33**	**90.68**	**45.20**	**73.62**
Nominal budget: 11.1%											
FastV	69.02	81.47	56.73	**2168.82**	44.22	2.12	1.19	1.45	82.02	2.14	52.06
SparseVLM	**70.04**	64.73	**58.30**	1596.47	47.50	4.15	2.90	3.82	74.91	4.89	48.65
Ours	62.06	**82.66**	53.32	1842.03	**75.06**	**15.48**	**12.74**	**14.23**	**83.89**	**19.13**	**59.61**

On Qwen2.5-VL, LearnPruner attains higher RelAcc. than our method at all three tiers: 94.42% versus 92.32%, 89.38% versus 83.86%, and 73.74% versus 64.53%. It also leads in both $\mathrm{rel}_{\mathrm{QA}}$ and $\mathrm{rel}_{\mathrm{VG}}$ at every tier. The gap widens as the budget becomes tighter, especially for grounding. Our method does not lead on this backbone, although it remains competitive with the other baselines.

B.3 Module Overhead: Training-Free vs. Learned Selector

On Qwen3-VL-8B, the FLOPs ratio is 0.2606 and prefill is accelerated by 1.28×, whereas decode speedup is 0.92×. Reducing visual tokens directly shortens the prefill sequence. The time-weighted combination of the two stages yields an end-to-end speedup of 0.95×, so the prefill gain is outweighed by increased decoding time. Estimated output lengths reconstructed from stored answers are similar across methods (see below), which makes longer outputs an unlikely main cause of the slower decoding, although these estimates are not generated-token counts. Overhead in the pruned execution path or KV-cache handling may contribute, but we did not isolate the cause. These numbers are benchmark end-to-end latencies under actual EOS termination.

Table A11: Visual-token pruning performance on Qwen2.5-VL. Units follow Table 1.

Method	TextVQA	POPE	GQA	MME	SQA	RefCOCO	RefCOCO+	RefCOCOg	$\mathrm{rel}_{\mathrm{QA}}$	$\mathrm{rel}_{\mathrm{VG}}$	RelAcc.
Dense baseline, 100% visual tokens											
Vanilla	77.93	88.76	61.37	2331.38	88.20	89.12	80.98	85.75	100.00	100.00	100.00
Nominal budget: 33.3%											
FastV	77.22	85.42	57.15	2264.82	51.41	58.80	53.05	60.92	88.78	67.51	80.80
LearnPruner	**77.72**	**88.21**	59.14	**2308.13**	**86.02**	78.12	**70.03**	**76.51**	**98.40**	**87.79**	**94.42**
Ours	73.94	87.68	**59.85**	2085.58	85.52	**79.19**	69.71	73.76	95.52	86.99	92.32
Nominal budget: 22.2%											
FastV	75.37	85.13	59.10	2311.02	45.22	28.26	24.15	28.17	87.86	31.46	66.71
SparseVLM	75.78	85.67	**60.29**	2212.91	48.29	32.55	27.94	32.77	88.33	36.41	68.86
GlimpsePrune	55.40	75.57	44.59	1346.88	75.01	0.37	0.10	0.40	74.34	0.34	46.59
VScan	74.45	85.98	57.91	2027.75	48.59	33.66	30.98	31.38	85.77	37.54	67.68
LearnPruner	**75.86**	86.47	57.96	**2321.87**	83.69	**69.07**	**60.99**	**67.33**	**96.74**	**77.11**	**89.38**
Ours	70.62	**86.92**	57.95	2006.84	**83.89**	62.23	55.64	58.49	92.83	68.92	83.86
Nominal budget: 11.1%											
FastV	64.71	80.90	54.83	2183.81	41.05	7.51	5.03	5.38	80.75	6.97	53.08
SparseVLM	**71.42**	75.97	**56.89**	1835.46	44.47	4.66	2.76	4.06	79.82	4.46	51.56
GlimpsePrune	58.14	34.37	38.68	1475.72	73.03	2.08	1.84	2.43	64.49	2.48	41.24
VScan	66.59	81.54	52.92	1797.33	44.77	8.07	5.80	6.87	78.28	8.08	51.95
LearnPruner	70.74	82.70	53.90	**2286.86**	80.57	**39.42**	**33.98**	**36.43**	**92.24**	**42.89**	**73.74**
Ours	62.24	**83.45**	53.47	1810.15	**80.91**	25.74	23.71	23.71	86.08	28.60	64.53

All inference latency is measured in FP16 with batch size 1, processing samples sequentially. We use greedy decoding (temperature $=$ 0, `do_sample=False`). For all methods, the `max_new_tokens` limit is 1 for POPE and 32 for the remaining benchmarks, with standard Hugging Face EOS-based early termination. Latency is timed per sample with CUDA synchronization before and after generation, averaged over all 5,023 RefCOCOg test samples without a separate warm-up, using the default Hugging Face attention implementation without FlashAttention. Output lengths are estimated by retokenizing each stored answer with its model tokenizer and adding one EOS token; they are reconstructed from stored answers, not counted from generated token IDs. The estimated mean over 5,023 RefCOCOg samples is 20.27 tokens for ours, 20.68 for LearnPruner, and 20.44 for dense Qwen3-VL-8B, and 23.60 and 23.88 for ours and LearnPruner on LLaVA-1.5. Only 4 samples for ours and 43 for LearnPruner reach the 32-token limit.

The main-text Table 7 compares the accuracy–efficiency trade-off of complete methods on RefCOCOg at the 11.1% tier: dense inference without pruning, the full LearnPruner method with its learned LPM classifier, and our full method (VTA+PDR). This is not a controlled comparison in which only the selector changes; the two methods also differ in their decoder-stage schedules and selection.

Our method yields higher RefCOCOg accuracy than LearnPruner on all three backbones: 4.90 versus 1.73 on LLaVA-1.5, 24.29 versus 9.85 on LLaVA-NeXT, and 53.99 versus 32.85 on Qwen3-VL-8B. Speed must be interpreted against both dense inference and LearnPruner. Relative to dense inference, prefill is faster on all three backbones and total speedup reaches 1.12× on LLaVA-1.5 and 1.39× on LLaVA-NeXT. Relative to LearnPruner, both total and prefill speedups are lower on all three backbones. On Qwen3-VL-8B, prefill speedup is 1.28× versus 1.95× for LearnPruner, and total speedup is 0.95× versus 1.04×, slightly below dense inference. The measured result is a trade-off in which higher accuracy comes with lower total speed than LearnPruner.

Measured peak memory and theoretical KV cache capture different costs and can move in opposite directions. Peak memory depends on the largest temporary token count over the schedule, while KV cache is a steady-state cost based on the layer-weighted mean retained count. On Qwen3-VL-8B, our method has a smaller KV cache (4.4 versus 7.0MB), reflecting LearnPruner's larger measured retention of 18.4%, and similar peak memory. On LLaVA-NeXT, LearnPruner is lower on both measures. On LLaVA-1.5, our method uses less peak memory, while LearnPruner has a slightly smaller KV cache. These measures need not rank methods identically.

Training cost: VTA vs. LearnPruner LPM. We also compare the training cost of the two learned selectors. VTA is trained entirely from pre-decoder features and therefore does not execute the LLM decoder during training. In contrast, LearnPruner trains its LPM using the next-token prediction loss

of a frozen 7B decoder, with gradients propagated through the frozen decoder to the LPM via a Straight-Through Estimator. We measure VTA without feature caching: the frozen vision encoder and projector compute visual features at every step. This mode is implemented for LLaVA-1.5, so the comparison uses that backbone. As shown in Table A12, VTA's estimated GPU-hour cost is approximately 2.1× lower. Both methods are trained on NVIDIA RTX A6000 GPUs with 46GB memory. VTA uses one GPU with a batch size of 128 and takes 7.05 seconds per step, measured over the first 170 steps including a validation pass; extrapolating to the five-epoch schedule of 3,900 steps gives an estimated full-schedule training cost of about 7 hours 40 minutes. LearnPruner uses four GPUs with a per-GPU batch size of 1 and takes approximately 3 hours 59 minutes.

VTA is trained on the 100k-sample subset for 5 epochs, corresponding to approximately 500k cumulative sample exposures. LearnPruner performs 5000 optimizer steps on a subset of approximately 62k samples with world size 4, per-GPU batch size 1, and a global effective batch of 16 through gradient accumulation, corresponding to about 80k sample exposures. Because the two methods are trained under different conditions, this is not a comparison of the cost of equal training amounts or of reaching equal performance.

Table A12: Training cost of VTA and the LearnPruner LPM on LLaVA-1.5, without feature caching. The VTA entry is an estimated full-schedule training cost extrapolated from the step time measured over the first 170 steps.

	Wall-clock	GPUs	GPU-hours
VTA	≈7h40m (5 epoch, est.)	1	≈7.7 (est.)
LearnPruner (LPM)	≈3h59m (5000 step)	4 (DDP)	≈16.0

For $D = 4096$ backbones (LLaVA-1.5 and Qwen3-VL-8B), VTA contains 8.94M parameters in two symmetric $4096 \to 512 \to 512 \to 4096$ bottleneck MLPs and occupies 35.8MB in FP32. The LearnPruner LPM contains 0.33M parameters in a $1024 \to 256 \to 256 \to 2$ classifier and occupies 1.3MB. VTA is about 27 times larger because it reconstructs a 4096-dimensional representation rather than producing two logits. Relative to a frozen 7B backbone, the modules account for about 0.13% and 0.005%. Both therefore have small storage and deployment costs.

Table A13 extends the comparison to FastV, VisPruner, and SparseVLM. FLOPs use the same layer-weighted method as the main-text Table 7. On LLaVA-1.5, all five pruned methods have FLOPs ratios between 0.2392 and 0.2420. On LLaVA-NeXT, VisPruner lies between LearnPruner and our method. Higher accuracy generally coincides with lower total and prefill speedup.

Table A13: Full module-overhead comparison on RefCOCOg at the 11.1% tier. FLOPs use the layer-weighted ratio from the main-text Table 7. Methods marked no module have no additional learned selector. Peak Mem is measured peak GPU memory in MB.

Backbone	Method	RefCOCOg	FLOPs	Speedup	Prefill	Peak Mem (MB)
LLaVA-1.5	No pruning (dense)	48.93	1.0000	1.00×	1.00×	15617.8
	FastV (no module)	0.46	0.2399	1.15×	2.28×	14354.0
	VisPruner (no module)	2.35	**0.2392**	1.13×	2.75×	14354.0
	SparseVLM (no module)	0.50	0.2397	0.88×	1.93×	18051.0
	LearnPruner (LPM)	1.73	0.2404	**1.17**×	2.45×	15016.4
	Ours (VTA+PDR)	**4.90**	0.2420	1.12×	2.09×	**14348.0**
LLaVA-NeXT	No pruning (dense)	80.41	1.0000	1.00×	1.00×	17011.9
	VisPruner (no module)	3.46	0.1426	**1.61**×	**5.71**×	**13647.3**
	LearnPruner (LPM)	9.85	**0.1392**	1.50×	4.61×	14594.1
	Ours (VTA+PDR)	**24.29**	0.1483	1.39×	2.65×	14989.3

B.4 Component Ablation

We isolate the contributions of the encoder prior and diversity using three variants at the 22.2% tier: diversity only, encoder prior only, and the full method that combines both components. Their effects vary across backbone families. On LLaVA-1.5, the full method performs best on TextVQA,

POPE, and RefCOCOg. It remains close to the best result on GQA, MME, and SQA, with differences of -0.03, -8.90, and -0.79, respectively. On LLaVA-NeXT, the full method performs best on TextVQA, POPE, GQA, and RefCOCOg. Its RefCOCOg score is 7.18 points above the encoder-prior-only variant. In contrast, diversity only is stronger on the Qwen family. It exceeds the full method on all six metrics for Qwen2.5-VL, including a 6.29 point difference on RefCOCOg. For Qwen3-VL-8B, the full method exceeds diversity only on POPE and RefCOCOg. Section B.5 examines this difference between the LLaVA and Qwen families. In all three variants, encoder-side selection is the same VTA+saliency score; only the decoder prior and diversity terms change.

Table A14: Ablation of the prior and diversity components at the 22.2% tier. Full is the adopted configuration that uses both components. All variants use the same VTA+saliency encoder selection.

Model	Variant	TextVQA	POPE	GQA	MME	SQA	RefCOCOg
LLaVA-1.5	+ diversity only	56.20	83.67	**60.35**	1802.17	**69.16**	15.83
	+ encoder prior	55.50	84.92	60.07	**1816.13**	68.57	16.32
	full (Ours)	**56.97**	**85.08**	60.32	1807.23	68.37	**18.99**
LLaVA-NeXT	+ diversity only	58.02	88.88	63.12	1805.86	**68.67**	29.68
	+ encoder prior	60.09	89.11	63.95	**1836.76**	68.57	40.14
	full (Ours)	**60.38**	**89.16**	**64.03**	1828.39	68.62	**47.32**
Qwen2.5-VL	+ diversity only	**72.41**	**87.18**	**59.01**	**2107.15**	**84.88**	**64.78**
	+ encoder prior	70.12	86.44	57.25	1965.34	84.23	56.42
	full (Ours)	70.62	86.92	57.95	2006.84	83.89	58.49
Qwen3-VL-8B	+ diversity only	**75.14**	88.03	**59.95**	**2125.40**	**87.85**	74.58
	+ encoder prior	73.08	88.33	59.05	2043.18	87.46	76.59
	full (Ours)	74.44	**88.35**	59.15	2110.47	86.91	**77.46**

B.5 Prior and T2V-Native Recall@K Analysis

Following Eq. 14 in Section 3.3, full PDR combines the decoder-native T2V signal with the encoder-stage VTA prior:

$$u_i^\ell = (1 - \gamma_\ell)\bar{a}_i^\ell + \gamma_\ell \bar{p}_i, \tag{18}$$

Here, $\bar{a}_i^\ell$ is the rank-normalized T2V cross-attention score at decoder layer ℓ, and $\bar{p}_i$ is the rank-normalized encoder-stage VTA prior. The γ_ℓ schedule has three stages (0.60, 0.35, 0.20) for LLaVA and five stages (0.60, 0.48, 0.35, 0.27, 0.20) for the Qwen family; the start and end values match, but the number of stages and intermediate values differ (Tables A4 and A7).

To examine how the two relevance signals differ across backbones, we independently measure the ground-truth box Recall@K of p_i and $a_i^{(\ell)}$ on the same visual-token set without encoder-stage pruning. We use 150 RefCOCOg test samples per backbone, drawn at random with seed 0, and measure the T2V score a_i at decoder layer 3 (0-indexed).

Table A15: Recall@K difference between the prior p_i and T2V-native score a_i. Positive values indicate higher recall for the prior. The prior is stronger for both backbones, but its advantage is generally smaller on Qwen2-VL than on LLaVA-1.5.

K	LLaVA-1.5 (prior − T2V)	Qwen2-VL (prior − T2V)
0.1	+0.085	+0.037
0.2	+0.104	+0.048
0.3	+0.090	+0.056
0.5	+0.036	+0.034
0.7	+0.034	+0.008

The VTA prior has higher Recall@K than the T2V-native signal at every K on Qwen2-VL. The prior is therefore not inherently weaker on this backbone. Its advantage over the T2V-native signal is smaller than on LLaVA-1.5, particularly at large K. This result accompanies the component ablation, where adding the prior does not consistently improve performance on some Qwen models.

One possible explanation is that we use γ_ℓ schedules with the same start and end values even though the relative quality of the two signals varies across backbones. A fusion weight chosen for LLaVA

may not have the same effect when the prior offers a smaller relative gain. Our experiments do not establish fixed γ_ℓ as the direct cause of the performance decrease. Backbone-specific ablations or adaptive weighting based on the relative reliability of the two signals are needed to test this explanation.

Architectural differences within the Qwen family may also matter. Qwen3-VL-8B uses DeepStack (Bai et al., 2025a) to reintroduce visual features into early decoder layers, whereas the other Qwen variants compared here do not use the same mechanism. This difference may affect how decoder-native relevance interacts with the encoder prior. We do not attribute the observed differences directly to DeepStack because we did not isolate its effect.

Overall, the relative strength of the encoder prior and the decoder-native T2V signal varies across backbones. The current analysis does not identify a single cause for the lower performance after adding the prior in some Qwen settings. In particular, the Qwen2-VL analysis provides an exploratory observation but does not establish the cause of the reversal on Qwen2.5-VL. We leave backbone-specific and adaptive fusion strategies for future work.

B.6 VTA vs. Saliency-Only: Encoder-Only Ablation

This ablation compares the learned aligner alone with training-free saliency alone using only encoder-stage scores. VTA uses α_v=0. The saliency baseline uses CLS-to-patch attention for LLaVA and the visual-token L_2 norm for Qwen with α_v=1. Decoder-stage PDR is not included. Table A16 shows mixed results. VTA is generally stronger on QA benchmarks such as TextVQA, MME, and SQA. RefCOCOg results vary across backbone families. VTA is substantially stronger on the two LLaVA backbones, scoring 28.85 versus 14.55 and 57.75 versus 50.63. Saliency alone is stronger on Qwen2-VL and Qwen2.5-VL, scoring 67.49 versus 41.75 and 83.83 versus 72.19. The two methods are nearly tied on Qwen3-VL-8B at 85.45 and 85.59. Both scorers use the adopted adaptive maxkeep selector, for which keep_ratio 0.5 is only the top-k lower bound, so the number of retained tokens can differ between scorers. On LLaVA-1.5 RefCOCOg, for example, VTA keeps 345 tokens on average and saliency keeps 288. With an equal 288-token budget (fixed top-k, Table A21, same checkpoint), VTA reaches 20.55 and saliency 14.55. This table therefore compares scorers under the adopted selector rather than at a fixed token count.

Table A16: Encoder-stage-only ablation comparing the learned aligner alone with training-free saliency alone, using the adaptive maxkeep selector with a top-k lower bound of $\mathrm{keep_ratio} = 0.5$, so the retained token count can differ between scorers. Decoder-stage T2V is not used. VTA uses the main checkpoints.

Model	TextVQA VTA / Saliency	POPE VTA / Saliency	GQA VTA / Saliency	MME VTA / Saliency	SQA VTA / Saliency	RefCOCOg VTA / Saliency
LLaVA-1.5	57.28 / 57.90	84.68 / 84.68	61.43 / 60.86	1793.62 / 1807.69	68.27 / 68.32	28.85 / 14.55
LLaVA-NeXT	60.10 / 60.88	88.92 / 88.90	64.42 / 64.19	1835.18 / 1828.72	68.77 / 69.16	57.75 / 50.63
Qwen2-VL	77.19 / 67.98	85.60 / 89.35	58.83 / 61.39	2285.87 / 2015.31	83.64 / 83.44	41.75 / 67.49
Qwen2.5-VL	76.89 / 68.50	86.25 / 88.80	58.05 / 60.65	2295.44 / 2152.03	85.97 / 85.82	72.19 / 83.83
Qwen3-VL-8B	77.82 / 74.92	90.55 / 90.24	60.38 / 61.29	2327.06 / 2256.76	93.11 / 92.86	85.45 / 85.59

B.7 Scorer Comparison at Identical Token Counts

Table A16 compares scorers only at the encoder stage, and the retained count can differ between scorers there. To isolate the contribution of the learned VTA under the same PDR, we change only the encoder score on Qwen3-VL-8B at the nominal 22.2% budget and evaluate all 5,023 RefCOCOg test samples. All arms reuse the per-sample encoder count of the reported VTA run, taking each scorer's own top-k, and share the same ratio-mode decoder, so every decoder stage keeps an identical number of tokens for every sample. Raw cosine uses the VTA module with its residual scale set to zero, i.e., cosine similarity of the frozen pre-decoder features. Re-running the VTA arm this way reproduces the reported outputs.

Acc@0.5 does not differ significantly among the three arms (McNemar $p = 0.87$ and 0.48). VTA yields only a higher mean IoU, by 1.40 points over saliency alone and 0.80 over raw cosine. Within the per-sample budget fixed to the VTA run, VTA's additional gain therefore appears in mean IoU

Table A17: Scorer comparison with per-sample token counts fixed to the VTA run (Qwen3-VL-8B, nominal 22.2% budget, all 5,023 RefCOCOg test samples). PDR is identical in all arms; its prior comes from each arm's encoder score. Differences are relative to VTA, with sample-bootstrap 95% confidence intervals in parentheses.

Encoder score	Acc@0.5	Mean IoU	ΔAcc (VTA − arm)	ΔIoU (VTA − arm)
Saliency only	77.56	67.45	-0.10 (-1.11, $+0.92$)	$+1.40$ ($+0.80$, $+2.03$)
Raw cosine + saliency	77.86	68.04	-0.40 (-1.41, $+0.66$)	$+0.80$ ($+0.15$, $+1.49$)
VTA + saliency (Ours)	77.46	68.85	–	–

(box precision) rather than in Acc@0.5. This experiment does not evaluate removing VTA's adaptive budgeting. This result covers one backbone, one budget, and one benchmark.

B.8 VTA Training-Objective Ablation: Reverse Loss, Dispersion, Residual Architecture

We test the reverse loss $\mathcal{L}_{t\to v}$, dispersion loss $\mathcal{L}_{\text{std}}$, and residual architecture on LLaVA-1.5 by changing one training component at a time. The adopted recipe uses λ_{tv}=0.5, λ_{std}=0.3, and a residual architecture. We compare it with four variants: plain removes the residual connection, lam0 sets λ_{tv}=0, std0 sets λ_{std}=0, and linear replaces the MLP with a single linear projection. All variants are trained on the same 100k-sample cache. The adopted residual recipe is the main checkpoint (five-epoch schedule, selected at step 200); plain uses a five-epoch schedule (step 300), and lam0, std0, and linear use a three-epoch schedule (step 200). All variants are selected by the same held-out matched-versus-mismatched gap, and every selected checkpoint falls within the first epoch. Because the criterion does not improve after the first epoch, all variants are compared at early checkpoints; the cosine learning-rate schedule makes the learning rate at step 200 differ by about 1.5% between the two settings. The linear variant uses one $4096 \to 4096$ linear map per modality with no hidden layer, nonlinearity, or residual connection (33.6M parameters versus 8.94M for the residual MLP), so it changes depth, parameter count, and the residual path together rather than isolating nonlinearity.

Table A18: Ablation of the VTA training objective and architecture using Recall@K on 500 RefCOCOg samples. The raw-cosine reference is the same as in Table 3.

Variant	K=0.1	K=0.2	K=0.3	K=0.5	K=0.7
residual (ours)	0.271	**0.455**	**0.598**	0.801	0.918
plain (no residual)	**0.273**	0.451	0.588	0.801	**0.929**
lam0 (no reverse loss)	0.257	0.442	0.591	**0.805**	0.925
std0 (no dispersion)	0.253	0.431	0.572	0.788	0.916
linear (no MLP)	0.082	0.124	0.173	0.357	0.600

Removing dispersion reduces recall by 0.002 to 0.026 across all values of K. At K=0.1, recall decreases from 0.271 to 0.253. The dispersion term improves separation between query-relevant regions and the background on Recall@K, an independent metric that is not directly optimized by the training objective. This experiment compares only the adopted λ_{std}=0.3 with its removal. It supports using the dispersion term but does not establish a monotonic benefit from increasing its weight.

The variant without reverse loss is not consistently weaker on Recall@K. This experiment therefore does not show an independent performance gain from $\mathcal{L}_{t\to v}$. We retain it as a symmetric regularizer in the bidirectional contrastive objective but do not attribute VTA's improvement to this term.

The residual and plain variants are effectively tied, differing by at most about 0.01 in recall at every K, so the results do not attribute VTA's gain to the residual connection itself. Replacing the nonlinear MLP with a single linear projection produces lower recall than raw cosine at every K and falls below the random value of 0.099 at K=0.1, reaching 0.082. Its stored checkpoint also has about half the $\cos_\text{std}$ of the residual and plain variants, indicating reduced score dispersion. The observed performance difference is thus more closely associated with nonlinear projection than with the residual connection.

We also measure the downstream accuracy of the same five variants on SQA, MME, TextVQA, and POPE using top-k selection with $\mathrm{keep_ratio} = 0.5$.

Table A19: Downstream accuracy of the five LLaVA-1.5 variants using top-k selection with $\mathrm{keep_ratio} = 0.5$. We report grounding-sensitive metrics on MME and TextVQA and less grounding-sensitive metrics on SQA and POPE.

Variant	SQA	MME	TextVQA	POPE
base(dense)	69.51	1867.96	58.21	85.24
residual (ours)	68.32	1775.73	56.53	84.36
plain	68.37	1795.23	56.60	83.88
std0	68.72	1766.40	54.40	84.72
linear	68.62	1643.67	53.82	82.16
lam0	68.67	1811.75	56.95	84.28

MME and TextVQA require localization of detailed visual regions such as OCR regions and specific scene elements. SQA can often be answered from text and coarse layout, while POPE concerns object presence over the full image rather than a specific region. On the more grounding-sensitive metrics, std0 decreases MME by 9.3 points and TextVQA by 2.13 points. The linear variant decreases them by 132 and 2.71 points. These changes agree with the Recall@K result. On the less grounding-sensitive metrics, std0 improves SQA by 0.4 points and POPE by 0.36 points. The linear variant changes SQA only slightly but reduces POPE by 2.2 points. Lam0 slightly improves MME, TextVQA, and SQA by 36, 0.42, and 0.35 points and slightly lowers POPE from 84.36 to 84.28. Since this is a single on/off comparison, it only shows that removing the reverse loss does not clearly hurt these metrics in this setting.

We directly apply the five variants with encoder-only top-k pruning and measure final RefCOCOg REC accuracy at two tiers. This is the actual box-match accuracy rather than Recall@K and uses all 5,023 test samples.

Table A20: RefCOCOg REC performance at $\mathrm{keep_ratio} = 0.20$ (tight token budget), encoder-only fixed top-k selection.

Variant	Acc@0.5	Mean IoU
Residual (Ours)	**4.82**	**17.61**
No reverse loss	4.38	17.31
Linear projection	4.36	17.29
No dispersion	4.32	16.99
No residual	4.08	17.06

Table A21: RefCOCOg REC performance at $\mathrm{keep_ratio} = 0.50$ (loose token budget), encoder-only fixed top-k selection (288 tokens for every variant).

Variant	Acc@0.5	Mean IoU
Linear projection	**20.98**	30.15
Residual (Ours)	20.55	**30.22**
No residual	20.23	29.45
No reverse loss	20.17	29.83
No dispersion	19.19	28.91

At the tighter $\mathrm{kr}{=}0.20$ tier, the residual variant performs best among the five variants. The ordering and margins of the other variants differ from Recall@K. Plain is nearly tied with residual on Recall@K, while linear is substantially weaker. In final REC accuracy, plain is the weakest variant and linear slightly exceeds the raw-cosine level (4.36 vs. 4.10). We therefore do not assume that the Recall@K ordering transfers directly to final performance. At $\mathrm{kr}{=}0.5$, linear slightly exceeds residual on Acc@0.5, while residual is only slightly higher on Mean IoU. The advantage of the residual architecture therefore depends on the token tier. It is largest under a tight budget and disappears or reverses on one metric under the looser budget. Nevertheless, removing dispersion remains worse than the full model at both budgets. This supports the role of dispersion in distinguishing relevant tokens and preserving final REC performance across token budgets.

B.9 Qualitative Grounding and QA Examples

Following Figure 4 in Section 5, we provide success and failure cases (Figure A1) and additional qualitative results below.

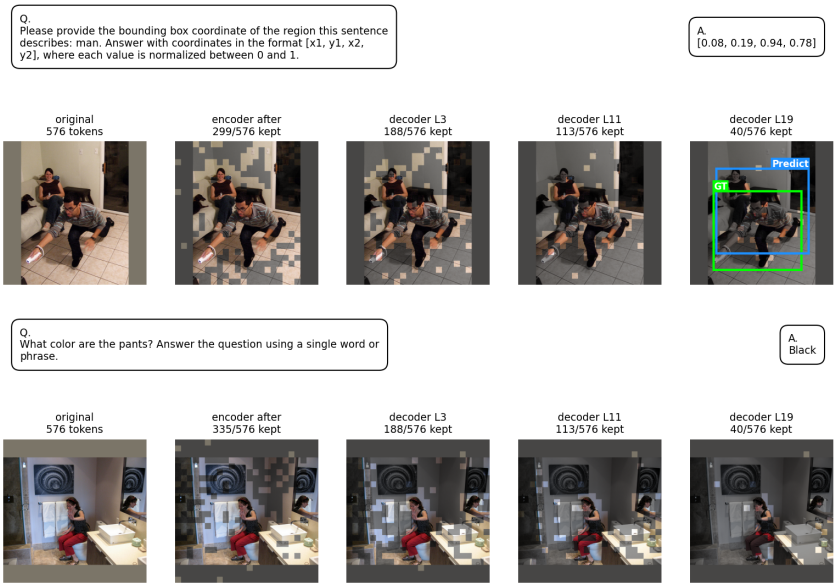


Figure A1: (Top) RefCOCO success: tokens focus on the queried man; predicted (blue) and ground-truth (green) boxes overlap. (Bottom) GQA failure: tokens favor the face/background over the pants, yielding "black" instead of "red".

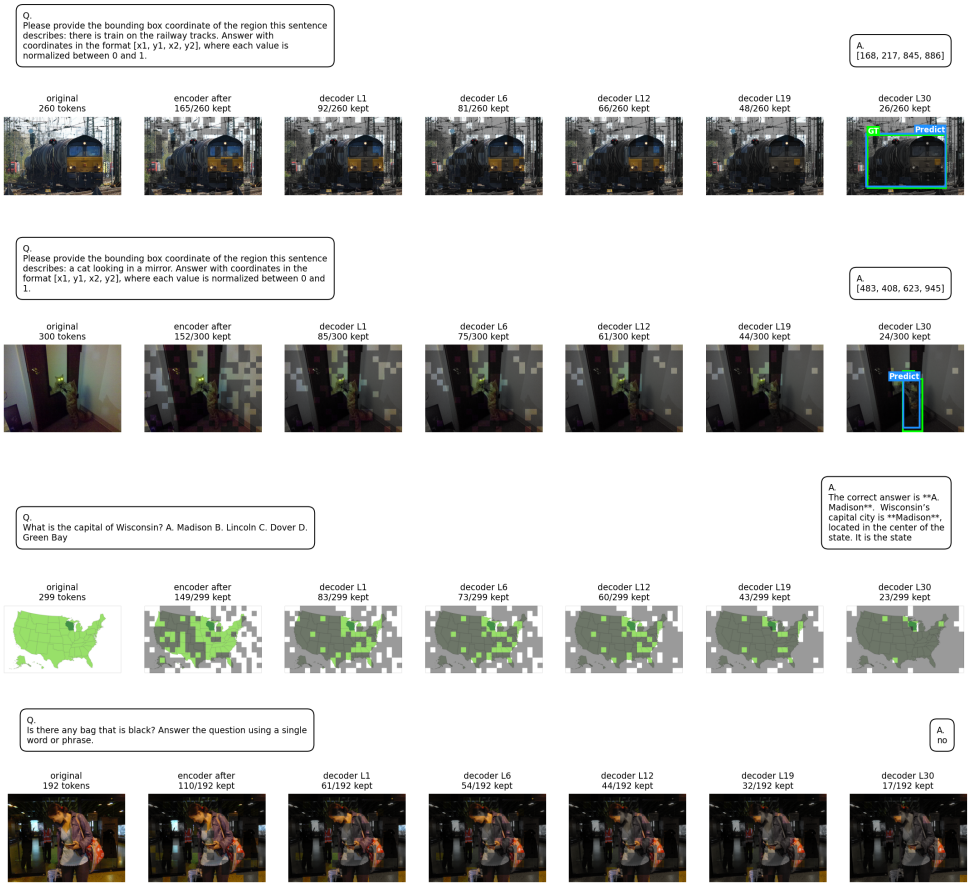


Figure A2: Additional qualitative results from top to bottom. For the train and the cat in the mirror on RefCOCOg, the retained tokens concentrate on the respective referent. They focus on Wisconsin in ScienceQA and on the queried black bag in GQA regardless of answer correctness.