VLVLM: LEARNING VARIABLE LENGTH VISUAL TOKENS FOR EFFICIENT VISION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Image tokens account for much of the compute cost of vision language models (VLMs), particularly at high resolution. Prior research has sought to reduce this cost by pruning tokens deemed redundant or uninformative, primarily through training-free approaches focusing on general visual understanding tasks. In this work, we show that these approaches incur substantial accuracy losses on text-rich images, where they are outperformed by a simple finetuning strategy that pools consecutive fixed-size groups of image tokens. However, fixed pooling is fundamentally limited in that it compresses all regions uniformly, despite differences in the amount of information they might contain. We introduce **VLVLM**, a training-based approach that dynamically chunks image tokens using a lightweight boundary predictor. At $4\times$ compression (i.e., using only 25% of the original image tokens), VLVLM retains 89% of the uncompressed models' performance on text-heavy visual benchmarks and outperforms both fixed pooling (87%) and training-free compression (82%), while preserving 99% performance on general tasks. Notably, the advantage persists under a more aggressive $8\times$ compression: we obtain 85% retention compared with 81% for fixed pooling. These gains hold across VLM families, image encoders, and language model sizes (up to 14B). Further analyses show that the learned boundaries are semantically meaningful, aligning closely with text-rich regions. We will release all our code and checkpoints upon acceptance.

1 INTRODUCTION

Vision language models (VLMs) typically convert images into discrete token sequences using vision encoders which are then fed into the language model. At high resolution or in case of videos, this process can produce (tens of) thousands of visual tokens, incurring substantial computational overhead and undermining both efficiency and scalability (Zhang et al., 2025d; Shang et al., 2026; Yu et al., 2026). Numerous approaches have been introduced in the literature to compress image token sequences, leveraging various heuristics based on attention patterns and similarity between visual and/or text representations (Liang et al., 2022; Bolya et al., 2023; Zhang et al., 2025d; Fang et al., 2026; Shang et al., 2026). These approaches are largely training-free, in that they compress visual representations *without adapting the language model to the compressed inputs*.

Our work is motivated by the observation that this mismatch between training and inference limits the preservation of finer-grained visual information in VLMs. Despite preserving good performance on general visual understanding benchmarks, training-free approaches incur substantial losses on tasks that rely on fine visual details such as understanding text-rich images and documents—capabilities that underpin most VLM use in the wild (Chou et al., 2025). Finetuning is a natural solution to this issue; in our experiments, we indeed find that a simple finetuning strategy with fixed pooling that aggregates every fixed number of consecutive visual tokens surpasses training-free approaches in performance, especially on benchmarks that rely heavily on optical character recognition (OCR). However, since information in an image is not uniformly distributed, fixed pooling still incurs noticeable performance degradation compared to the uncompressed model.

We introduce **VLVLM** (**V**ariable-**L**ength Visual Tokens for Efficient **V**ision **L**anguage **M**odels), a new VLM architecture that groups consecutive image tokens into variable-length chunks (see Figure 1; §2). Inspired by recent work on learnable tokenization for language modeling (Nawrot et al., 2023; Hwang et al., 2025), where the non-uniform distribution of information is a well-known

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.

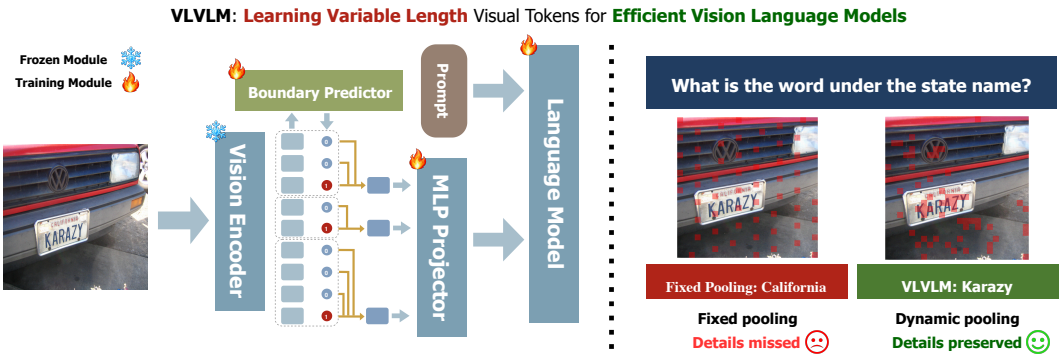


Figure 1: VLVLM architecture. Our proposed boundary predictor outputs a boundary mask that divides tokens into boundary and non-boundary tokens, where each group of consecutive non-boundary tokens is merged together with the subsequent boundary token. Our method outperforms baselines on text-centric visual tasks.

problem, we use a lightweight boundary predictor that takes each patch from the vision encoder and predicts whether it marks the end of a chunk. We average-pool the representations within each chunk to produce a single vision token, yielding a shorter sequence for the language model. We train the boundary predictor with an auxiliary loss that controls the token compression rate.

In our experiments with the LLaVA architecture (Liu et al., 2023; 2024a), we find that VLVLM shows better performance than strong training-free and training-based baselines at compression rates of $4\times$, $8\times$, and $10\times$, with the largest gains in text-rich visual understanding tasks (§4). These gains extend to stronger image encoders such as SigLIP2 (Tschannen et al., 2025), and also to a stronger underlying language model, Qwen-2.5 14B (Qwen et al., 2025). Our approach consistently offers the best accuracy–FLOPs trade-off among all evaluated methods; it learns semantically meaningful token boundaries and remains compatible with complementary efficiency techniques, including FlashAttention and token compression within the LLM (§5).

2 VLVLM Architecture

Preliminaries. VLMs typically take one or more images or videos and a text prompt as input and predict text as output. We limit our experiments to single image settings but the methods are generalizable. The input image $\mathbf{X} \in \mathbb{R}^{3\times H\times W}$ with is first converted to a sequence of patches of length L, which are encoded by a transformer encoder (ViT; Dosovitskiy et al., 2021). We refer to these encoded features as "image tokens". The tokens $\mathbf{Z} \in \mathbb{R}^{L\times D_{\text{img}}}$ are then fed into a multi-layer perceptron (MLP) projector to align with text token representations, $\mathbf{H} \in \mathbb{R}^{L\times D_{\text{LM}}}$, which are input to the language model along with the token embeddings of the text prompt.

General Structure. Image patch sequences incur a substantial computational overhead. For example, with a patch size of 16×16 pixels, a single 1024×1024 pixel image yields 4096 image tokens, even before accounting for the text. We aim to reduce the number of image tokens to $L' = \phi \cdot L$, where ϕ is a predefined compression ratio. As illustrated in Figure 1, we insert a lightweight boundary predictor between the vision encoder and the MLP projector. The boundary predictor predicts a binary output for each patch determining whether a chunk ends with this patch. For each consecutive patch sequence with boundary labels $(0, \ldots, 1)$, we average their individual representations to obtain $\mathbf{Z}' \in \mathbb{R}^{L'\times D_{\text{img}}}$, which are fed into the MLP projector.[1]

Boundary Predictor. The boundary predictor is implemented as a two-layer MLP with a sigmoid head. For representation z_i at position i, it predicts a boundary probability p_i. During training, we obtain the exact boundary decisions, $b_i \in \{0, 1\}$, by sampling from a Bernoulli distribution defined by p_i. To keep the network differentiable, we adopt the Gumbel-Sigmoid reparameterization trick

[1] In early experiments, other pooling methods performed worse (e.g., using only the boundary patch).

along with the straight-through estimator following Nawrot et al. (2023), who introduced learnable segmentation in text models. During inference, we select a subset of image tokens with the highest boundary probabilities such that the compression ratio matches the specified compression budget. Further details of the methodology along with detailed algorithms are provided in Appendix B.

Boundary Loss. Without constraints, the boundary predictor collapses to a degenerate solution during training (either all boundaries or no boundaries). To prevent this, in addition to the VLM finetuning cross-entropy objective, $\mathcal{L}_{\text{CE}}$, we introduce an auxiliary loss that encourages the boundary predictor to classify $(100 * \phi)\%$ of image patches in a batch as boundaries, where ϕ is the predefined compression ratio. Formally, consider the boundary predictions for a batch $\boldsymbol{b} \in \{0,1\}^{B\times L}$. We compute the number of boundaries per instance: $k_i = \sum_{l=1}^{L} \mathrm{b}_{i,l}$. Assuming a Binomial distribution $K \sim \text{Binomial}(L, \phi)$, the boundary loss is defined as the negative log-likelihood of the number of boundary patches under this distribution (Nawrot et al., 2023),

$$\mathcal{L}_{\text{boundary}} = -\frac{1}{BL}\sum_{i=1}^{B} \log\left[\binom{L}{k_i}\phi^{k_i}(1-\phi)^{L-k_i}\right]. \tag{1}$$

Intuitively, this loss discourages deviations from the expected number of boundaries, $L \cdot \phi$, pushing the model to preserve a stable compression ratio. Setting ϕ to 1, this loss will be minimized when all patches are predicted as boundaries; setting it to 0 will result in no boundaries being predicted. The final training loss is: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{boundary}}$.

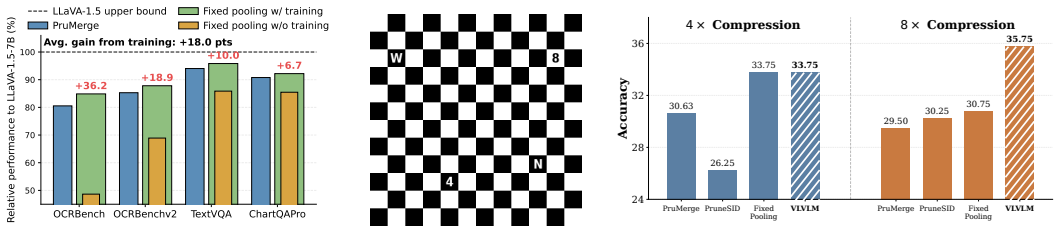
3 Experimental Setup

Finetuning Recipe. We conduct our main experiments with the LLaVA-1.5 SFT dataset containing 665K examples (Liu et al., 2024a), adopting their finetuning setup and hyperparameter configuration: hyperparameter and training details are presented in Appendix A. Our language backbone is vicuna-1.5-7B (Zheng et al., 2023), and we use CLIP ViT-L-14 (Radford et al., 2021) as the image encoder with an input resolution of 336×336. In the token alignment or "pretraining" phase (Liu et al., 2023), we train only the MLP projectors and the boundary predictor while freezing all other components; in the subsequent instruction finetuning phase (Liu et al., 2023), we freeze the image encoder and update all remaining modules. To validate if our results can generalize beyond CLIP encoders, we also conduct experiments with a different image encoder, SigLIP2, where we use the pretrained SigLIPv2 ViT-L model (Tschannen et al., 2025) with patch size of 16. To understand the impact of the backbone language model, we also train a variant with Qwen2.5-14B-instruct (Qwen et al., 2025).

Baselines. We compare VLVLM with the original **uncompressed model**, two training-free baselines, and two training-based baselines. For the training-free methods, we include two state-of-the-art methods: **LLaVA-PruMerge** (Shang et al., 2026) which leverages both class-attention and cosine similarity to cluster the image features produced by ViTs, and **PruneSID** (Fang et al., 2026) which uses low-rank PCA decomposition to prune redundant image tokens. For the training-based methods, we include **Perceiver** (Jaegle et al., 2021), adopted in BLIP2 (Li et al., 2023a) and Llama3 (Grattafiori et al., 2024), which uses a cross-attention module to project a long sequence to a fixed-sized set of latents, and **fixed pooling**, which merges tokens by aggregating every fixed number of consecutive tokens in the one-dimensional sequence. We evaluate all compression methods, including VLVLM, under three compression settings: $4\times$ ($\phi = 0.25$), $8\times$ ($\phi = 0.125$), and $10\times$ ($\phi = 0.1$).

Evaluation. Text-centric benchmarks are specifically designed to assess how well VLMs handle text perception, recognition, and comprehension within images; in this work, we adopt TextVQA (Singh et al., 2019), OCRBench (Liu et al., 2024d), OCRBench v2 (Fu et al., 2025b), DocVQA (Mathew et al., 2021), and ChartQAPro (Masry et al., 2025). We also evaluate our method on nine general (i.e., not text-specific) VQA benchmarks. Further details can be found in Appendix G. For all experiments, we apply McNemar's test (McNemar, 1947) to assess whether the observed performance improvements over the baselines are statistically significant.[2] For efficiency evaluation, we use fvcore (Facebook, 2019) to measure TFLOPs (single forward pass with a batch size of 1). In the Appendix C, we present

[2]McNemar's test is not applicable for free-response benchmarks (LLaVA-B (Liu et al., 2023), MM-Vet (Yu et al., 2024)) and benchmarks without public ground truth (VQAv2 (Agrawal et al., 2016)).



(a) Results on training the LLM with simple fixed pooling.

(b) Example from synthetic dataset.

(c) VLVLM performs best on synthetic tasks, especially for 8× compression.

Figure 2: Finetuning with fixed pooling allows the model to largely recover performance and eventually surpass training-free methods on OCR-heavy tasks. Our controlled synthetic-data evaluation shows that fixed pooling cannot adapt to the image content, leading to suboptimal results.

more detailed efficiency analysis regarding FLOP-performance tradeoff (Figure 7), reduced memory usage (Figure 8), and training speedup due to flash-attention (Table 8).

4 Results and Analyses

Table 1: Results on LLaVA 1.5 7B. The best and second best results are highlighted among training-free and training-based methods. We denote with an asterisk (*) statistically significant improvements relative to fixed pooling. Avg. Ret. indicates average retention in performance across tasks compared with the uncompressed baseline. Detailed results for general visual understanding tasks (others) can be found in Appendix Table 9.

Model	TextVQA	OCRBench	OCRBenchv2	DocVQA	ChartQAPro	Avg. Ret. (OCR)	Avg. Ret. (others)	TFLOPs
upper bound (keep 576/576 tokens out of the image encoder)								
LLaVA-1.5-7B	58.16	18.5	23.8	25.61	12.04	100%	100%	8.89
4× compression (keep 144/576 tokens out of image encoder)								
PruMerge	54.69	14.9	20.3	16.37	10.93	82.91%	98.14%	3.18(2.8×)
PruneSID	53.03	14.5	19.4	13.40	10.38	77.92%	96.06%	3.16(2.8×)
Perceiver	53.10	13.4	19.1	13.83	10.18	76.50%	94.10%	3.17(2.8×)
fixed pooling	55.74	15.7	**20.9**	18.78	**11.10**	86.80%	**99.16%**	3.16(2.8×)
VLVLM	**55.84**	**16.5**	**20.9**	**20.39***	10.91	**88.65%**	99.05%	3.18(2.8×)
8× compression (keep 72/576 tokens out of the image encoder)								
PruMerge	54.46	14.6	18.4	14.28	10.48	78.53%	94.31%	2.22(4.0×)
PruneSID	53.14	13.3	18.6	12.78	10.62	75.90%	95.30%	2.21(4.0×)
Perceiver	52.78	13.2	19.0	14.24	9.98	76.08%	95.13%	2.21(4.0×)
fixed pooling	54.05	14.3	19.5	16.00	10.81	80.88%	93.82%	2.21(4.0×)
VLVLM	**55.05***	**15.6**	**20.1**	**17.59***	**10.94**	**84.59%**	**96.79%**	2.23(4.0×)
10× compression (keep 58/576 tokens out of the image encoder)								
PruMerge	**54.01**	13.7	17.9	13.76	10.09	75.93%	92.30%	2.02(4.4×)
PruneSID	52.59	13.1	18.6	12.33	10.42	74.81%	94.43%	2.02(4.4×)
Perceiver	52.47	13.6	19.2	13.81	10.32	76.81%	94.36%	2.01(4.4×)
fixed pooling	53.40	13.9	19.1	15.32	**11.25**	80.09%	**95.54%**	2.01(4.4×)
VLVLM	53.87	**14.6**	**19.8**	**16.60***	10.65	**81.60%**	95.07%	2.04(4.4×)

4.1 Main Results

Motivating Experiments. To examine the roles of finetuning and variable-length patches, we conduct two experiments: we first compare fixed pooling with and without finetuning, then evaluate fixed and VLVLM on a controlled task requiring the preservation of localized visual details. Figure 2(a) compares fixed pooling with and without finetuning with PruMerge at 4× compression. Finetuning substantially improves fixed pooling performance across all four benchmarks, outperforming PruMerge. These results highlight the importance of training the LLM to make use of compressed visual tokens. To examine the limitations of uniform compression, we synthesize a controlled task,

Table 2: Results on LLaVA 1.5 7B with SigLIP2 image encoder. Detailed results for general visual understanding tasks (others) can be found in Appendix Table 10.

Model	TextVQA	OCRBench	OCRBenchv2	DocVQA	ChartQAPro	Avg. Ret. (OCR)	Avg. Ret. (others)	TFLOPs
upper bound (keep 576/576 tokens out of the image encoder)								
SigLIP2 ViT	60.47	36.2	36.9	28.13	11.70	100%	100%	8.89
4× compression (keep 144/576 tokens out of image encoder)								
Perceiver	57.29	29.8	32.2	18.26	10.16	83.22%	94.73%	3.16(2.8×)
fixed pooling	**58.73**	**31.8**	**33.0**	**21.55**	10.69	**88.47%**	**97.36%**	3.16(2.8×)
VLVLM	57.88	31.1	32.5	20.71	**11.12**	87.67%	96.78%	3.18(2.8×)
8× compression (keep 72/576 tokens out of the image encoder)								
Perceiver	55.96	29.6	32.0	17.34	10.75	82.91%	94.84%	2.21(4.0×)
fixed pooling	**56.73**	29.2	30.9	17.51	**11.18**	83.20%	95.03%	2.21(4.0×)
VLVLM	56.72	**31.1***	**32.2**	**20.54***	10.81	**86.47%**	**95.70%**	2.23(4.0×)
10× compression (keep 57/576 tokens out of the image encoder)								
Perceiver	56.46	28.3	**30.9**	16.84	10.89	81.64%	94.85%	2.01(4.4×)
fixed pooling	55.99	30.3	29.9	16.97	10.61	81.72%	94.81%	2.01(4.4×)
VLVLM	**57.33***	**30.5**	**30.9**	**19.08***	**10.93**	**84.81%**	**95.95%**	2.04(4.4×)

in which the information needed to answer a question is confined to a small part of an otherwise repetitive image. Each image contains a black-and-white checkerboard with one randomly sampled character placed in a random cell within each quadrant (Figure 2(b)). For each image, we construct four questions asking the model to identify the character in each quadrant. Figure 2(c) shows that our VLVLM matches fixed pooling at $4\times$ compression but outperforms it along with all other baselines at $8\times$ compression. This suggests that adaptive boundaries can better preserve localized finer-grained information in images. See details of the synthetic dataset in Appendix A.

Results on LLaVA-1.5 7B. We summarize our main results for text-intensive tasks and average performance on other tasks in Table 1. Detailed performance breakdowns for the other tasks can be found in Table 9 in the Appendix. Overall, VLVLM outperforms all baselines on the majority of tasks. Fixed pooling consistently outperforms training-free methods (PruMerge and PruneSID) and the training-based Perceiver. On OCR benchmarks, dynamic patching consistently outperforms fixed pooling across all compression ratios, retaining 89% performance compared to the uncompressed LLaVA model under moderate compression ($4\times$) and up to 85% performance under aggressive compression ($8\times$ and $10\times$), outperforming both fixed pooling ($4\times$: 87%, $8\times$: 81%) and training-free compression ($4\times$: 82%, $8\times$: 78%). On coarse-grained VQA benchmarks, VLVLM still well preserves its performance, reaching more than 99% performance retention under $4\times$ compression. Detailed benchmark-level results are presented in Table 9 in the Appendix.

Impact of Vision Encoder. To examine whether VLVLM's benefits extend beyond the CLIP encoder used in the previous experiment, we replace it with SigLIP2 and retrain the models at three compression rates. As shown in Table 2, this replacement improves the uncompressed model's performance on four of the five text-rich benchmarks, with particularly large gains on OCRBench and OCRBench v2. With this stronger baseline, VLVLM slightly trails fixed pooling at $4\times$ compression (87.7% vs. 88.5% performance retention), but outperforms it by more than 3 points at $8\times$ and $10\times$.

Impact of the Underlying Language Model. To examine whether VLVLM's benefits extend to a different/larger language model, we replace Vicuna-7B with Qwen2.5-14B-Instruct while retaining the CLIP vision encoder. This replacement expectedly improves uncompressed performance across all five text-rich benchmarks (Table 3) while increasing TFLOPs due to its larger size (from 8.9 to 17.7). VLVLM achieves the highest average performance retention on both text-rich and general visual understanding benchmarks at all three compression rates.

Adapting an Already Trained VLM. We next examine whether VLVLM can be integrated into an existing VLM through limited continued finetuning. We transfer the learned boundary predictor from our SigLIP2 experiments to Qwen3VL-4B[3], introducing a $4\times$ token compression,

[3]In Qwen3VL, after image encoding, every 2×2 group of neighboring tokens are shuffled into a single "mega-token"; we let the transferred boundary predictor assign boundary probabilities to original image tokens before shuffling, and mean pool the boundary probabilities after shuffling, in which "mega-tokens" are downsampled.

Table 3: Results on LLaVA 1.5 with Qwen2.5-14B-instruct LLM. Detailed results for general visual understanding tasks (others) can be found in Appendix Table 11.

Model	TextVQA	OCRBench	OCRBenchv2	DocVQA	ChartQAPro	Avg. Ret. (OCR)	Avg. Ret. (others)	TFLOPs
upper bound (keep 576/576 tokens out of the image encoder)								
LLaVA Qwen2.5-14B	61.25	19.3	29.1	29.44	13.45	100%	100%	17.71
4× compression (keep 144/576 tokens out of image encoder)								
PruMerge	56.95	14.9	24.6	17.19	12.56	81.29%	94.80%	5.62(3.2×)
PruneSID	56.28	14.7	24.4	15.77	11.66	78.43%	95.44%	5.60(3.2×)
Perceiver	55.11	13.5	23.8	13.89	11.84	75.38%	92.47%	5.60(3.2×)
fixed pooling	**59.06**	17.0	26.2	21.48	**12.91**	88.69%	96.31%	5.60(3.2×)
VLVLM	58.76	**18.5**	**26.6**	**23.12***	12.89	**91.51%**	**98.86%**	5.63(3.2×)
8× compression (keep 72/576 tokens out of the image encoder)								
PruMerge	56.23	13.9	23.1	15.29	12.29	77.30%	91.91%	3.61(4.9×)
PruneSID	56.10	14.0	23.6	14.86	12.48	77.69%	94.24%	3.58(4.9×)
Perceiver	53.28	11.0	22.6	10.86	11.51	68.82%	90.81%	3.58(4.9×)
fixed pooling	55.86	14.8	24.9	17.14	12.90	81.51%	93.54%	3.58(4.9×)
VLVLM	**57.73***	**16.9***	**25.9**	**20.56***	**13.68**	**88.47%**	**95.34%**	3.61(4.9×)
10× compression (keep 58/576 tokens out of the image encoder)								
PruMerge	56.29	13.8	22.9	13.77	12.19	75.90%	91.04%	3.18(5.5×)
PruneSID	55.30	12.8	23.5	14.37	11.93	74.97%	94.05%	3.16(5.5×)
Perceiver	48.57	2.7	14.4	8.54	11.72	51.78%	73.26%	3.16(5.5×)
fixed pooling	55.92	13.5	24.5	15.96	**13.12**	79.43%	93.17%	3.16(5.5×)
VLVLM	**56.45**	**14.4**	**24.6**	**17.18***	12.02	**79.80%**	**94.57%**	3.22(5.5×)

and continue finetuning on 65K examples, approximately 10% of LLaVA-665K dataset. As shown in Table 4, under the same finetuning budget, VLVLM outperforms fixed pooling on TextVQA and OCRBench, respectively, while obtaining slightly lower scores on OCRBench v2 and ChartQAPro. On average, VLVLM reaches 75.80% performance retention of uncompressed Qwen3VL, surpassing both PruneSID (74.85%) and fixed pooling (73.49%).

Table 4: Results on continued finetuning on Qwen3VL.

Model	TextVQA	OCRBench	OCRBenchv2	ChartQAPro	Avg. Ret.
upper bound					
Qwen3VL-4B	77.99	**85.5**	**68.4**	26.57	–
+ SFT @ 65K	**81.49**	83.9	67.2	**27.89**	100%
Additional 4× compression					
PruneSID	71.23	49.9	46.7	**23.16**	74.85%
fixed pooling@65K	70.50	51.1	**51.3**	19.58	73.49%
VLVLM@65K	**76.20***	**54.6***	51.2	19.08	**75.80%**

These results provide initial evidence that the benefits of learned boundaries can transfer to an already trained VLM with limited additional finetuning. Because Qwen3VL processes images at their native resolution without resizing, the efficiency improvements are more pronounced: for an image with more than 3700 patches, TFLOPs drop from 40.3 to 17.4 with 4× compression. However, performance gaps remain relative to the uncompressed model, which we speculate could be closed with a larger training budget.

4.2 Boundary Analysis and Evaluation

To understand the quality of the predicted boundaries, we first quantitatively measure how much text within the images is "covered" by the boundaries (see Figure 3 and Table 5). We also ablate the boundary predictor MLP architecture and test alternative methods (see Figure 4). In Appendix E, we provide qualitative evaluation, including reconstruction quality after masking non-boundary tokens (Figure 10), alignment with CLS-token attention (Figure 11). We also provide qualitative analyses of failure modes of VLVLM in Appendix Figure 9.

Quantitative Evaluation of Predicted Boundaries. To quantitatively evaluate the boundaries discovered by VLVLM, we measure how much text is "covered" by the predicted boundaries and compare it to those identified with fixed pooling and PruneSID.

We first use a pretrained CRAFT model (Baek et al., 2019), which achieves more than 95% accuracy on ICDAR13 (Karatzas et al., 2013), to pinpoint text regions in images in the form of bounding boxes in pixel space. We then compute how much of the bounding box region and boundaries from VLVLM overlap, i.e., what percentage of text is covered by boundaries.

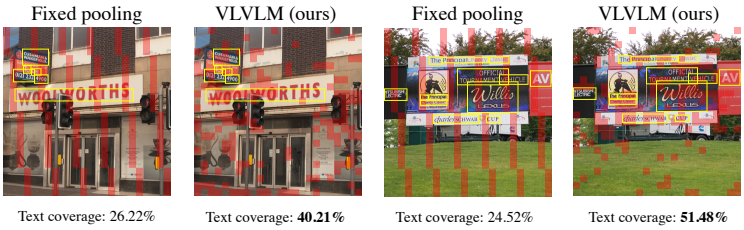
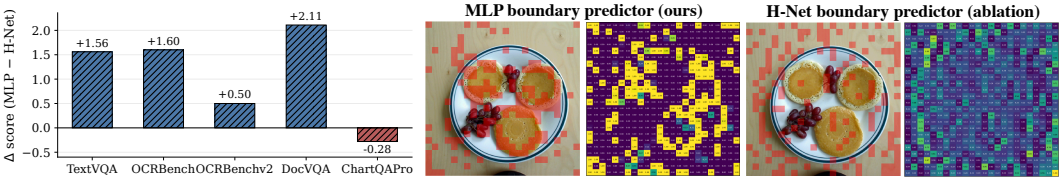


Figure 3: VLVLM reaches significantly higher text coverage compared to fixed pooling. We mark *boundary tokens* with red squares for both fixed pooling and VLVLM to illustrate how each method partitions the image.

Figure 3 shows example images highlighting text predicted by the craft model in bounding boxes along with the two metrics for VLVLM and fixed-pooling. Table 5 reports average text coverage across five text-intensive benchmarks. Our method reaches much higher average text coverage[4] than fixed pooling. Even though PruneSID reaches the same and sometimes even better text coverage, its task performance on text-intensive benchmarks suffers due to not training the language model itself.

Table 5: Average Text Coverage (%) for all five OCR benchmarks.

Method	Comp.	TextVQA	OCRBench	OCRBenchv2	DocVQA	ChartQAPro
PruneSID	4×	44.14%	**34.85%**	36.74%	**39.93%**	**42.22%**
Fixed	4×	25.32%	25.06%	25.01%	24.77%	25.48%
VLVLM	4×	**44.22%**	31.53%	**36.86%**	33.12%	41.02%
PruneSID	8×	30.33%	**21.87%**	**22.85%**	**28.01%**	**28.24%**
Fixed	8×	12.64%	12.53%	12.65%	12.44%	13.04%
VLVLM	8×	**32.41%**	16.87%	17.10%	15.86%	20.92%
PruneSID	10×	25.88%	**18.31%**	**19.64%**	**23.83%**	**24.68%**
Fixed	10×	9.73%	9.62%	9.78%	9.75%	9.80%
VLVLM	10×	**28.51%**	15.11%	14.63%	14.59%	19.87%



(a) Quantitative comparison.

(b) Boundary visualization.

Figure 4: Our MLP-based boundary predictor produces more semantically aligned boundaries and outperforms the H-Net strategy on OCR benchmarks.

Impact of Boundary Predictor Architecture. In our proposed method, the boundary predictor is implemented as a lightweight 2-layer MLP. Other strategies for boundary prediction have also been explored, primarily to tokenize text. For instance, H-Net (Hwang et al., 2025) relies on adjacent tokens' cosine similarity: the less similar two tokens are, the higher the probability that a boundary is placed between them. As illustrated in Figure 4, our MLP-based boundary predictor produces more confident decisions and yields boundaries that better align with semantic structure compared to H-Net. Furthermore, the results in Figure 4 also confirm that the H-Net strategy leads to overall inferior performance, particularly on OCR benchmarks.

5 Discussion

Can we Train with Fewer Samples? To understand the impact of finetuning data size, we train VLVLM models with 25%, 50%, and 75% of the 665K SFT samples (see Figure 5). Compared to

[4]All improvements against fixed pooling are proven to be statistically significant via Paired `t-test` with p-value < 0.05.

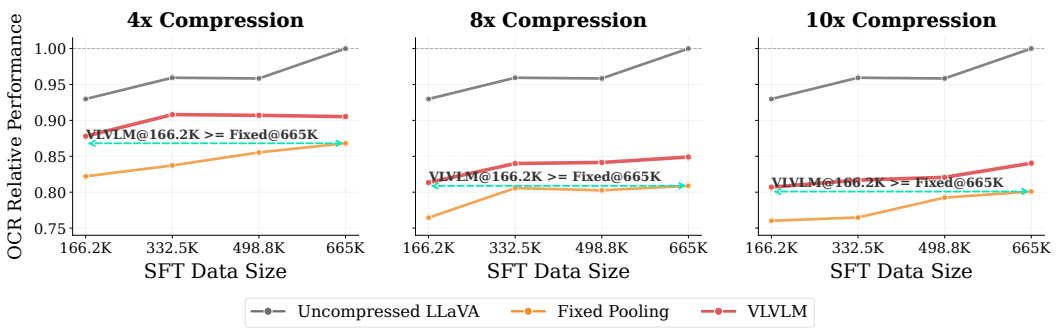


Figure 5: VLVLM performance with different data budgets. Our method outperforms fixed pooling at every single data size is more data-efficient.

fixed pooling,VLVLM outperforms under every data budget. Crucially, our approach is more *data efficient*: VLVLM achieves the same level of performance retention while using only 25% of the data required by fixed pooling (VLVLM @ 166.2K v.s. fixed 665K). Also, at higher compression rates (8 and 10×), VLVLM's performance continues to increase with more data, indicating that more finetuning data may lead to further performance improvements.

"You Only Train it Once". We test the model's behavior when the compression ratio at inference does not match that of training. As shown in Figure 6, on the left, we show the performance of VLVLM relative to the original LLaVA model; on the right, we illustrate the performance gap from matched-budget training. We observe at most 2% average performance drop in cases of compression ratio mismatch, highlighting its potential of simply *training the model once* and applying it to any arbitrary compression ratio at inference time. More importantly, owing to negligible performance drop, we can train a model much faster by using a more aggressive compression (e.g., 8× or 10×), and use a more moderate compression (e.g., 4× or even 2×) at inference time to reach better performance if desired.

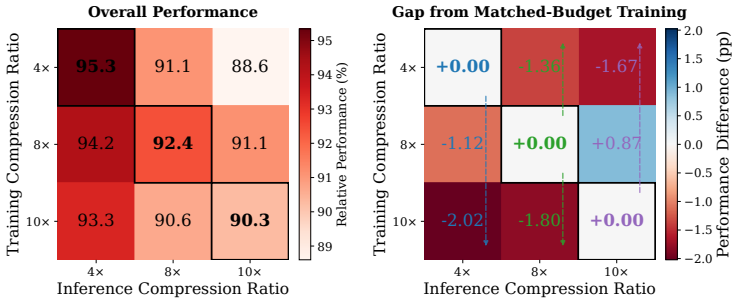


Figure 6: VLVLM supports changing the compression budget at inference with minimal change in task performance.

Combining with LLM-Stage Compression. VLVLM operates at the visual encoder stage. Prior work has also explored VLM compression at the LLM-stage. We investigate if they can be combined. We test PyramidDrop (Xing et al., 2025), which progressively drops image tokens in LLM layers. We compare VLVLM at 8× with VLVLM at 4× combined with PyramidDrop at 2×. This comparison makes sure the same number of image tokens are retained in the prefill stage. Table 6 shows that under a slightly increased TFLOPs (since LLM-stage compression is "late" compared to encoder-stage compression), combining VLVLM and PyramidDrop can even lead to stronger performance.

Table 6: VLVLM (encoder-stage compression) combined with PyramidDrop (Xing et al., 2025) (LLM-stage compression). Results shown at 8× compression.

Model	TFLOPs.	MME	TextVQA	OCRBench	OCRBenchv2	DocVQA	ChartQAPro
w/o Pdrop	2.23	1352.07	55.05	15.6	20.1	17.59	**10.94**
w/ Pdrop	2.60	**1379.60**	**55.95**	**16.0**	**20.7**	**19.94**	10.66

6 Related Work

Visual Token Compression. Early work in this direction focused on compressing within vision transformers for vision-only tasks like image classification. For instance, EViT (Liang et al., 2022) and ToME (Bolya et al., 2023) leverage attention and similarity measures to reduce redundant tokens. The major computational bottleneck in VLMs is the language model (owing to its larger size), not the vision encoder. Since training VLMs can be expensive, most work on improving VLM efficiency has focused on approaches which can be applied at inference. Many of the training-free approaches exploit various attention patterns to select "important" image tokens, that serve as centroids around which other tokens are clustered and merged (Shang et al., 2026; Hu et al., 2026; Zhu et al., 2026; Yang et al., 2026; Zhang et al., 2025b; Arif et al., 2024). In addition to inferior performance on text-rich visual tasks (as we show in our experiments), reliance on attention patterns can make these methods incompatible or less compatible with efficient attention implementations (FlashAttention, Dao et al., 2022b). Other training free approaches that avoid using attention patterns have also been explored (Fang et al., 2026; Ma et al., 2026; Wang et al., 2025b; Xu et al., 2026; Deng et al., 2025; Jiang et al., 2025; Alvar et al., 2025). Compressing the visual modality in isolation can create a representation mismatch with the language model affecting VLM performance (as we show in §4): therefore, recent works have also investigated leveraging the text prompt to guide the image compression process (Zhang et al., 2025d; Yu et al., 2026; Zhang et al., 2025c; Cai et al., 2025).

Training-based compression has also gained traction in recent work. Bai et al. (2025); Wang et al. (2025a); Marafioti et al. (2025) "merge" fixed 2×2 group of patches. Perceiver resampler has been employed in many VLMs (Li et al., 2023a; Grattafiori et al., 2024) to aggregate features in a fixed number of learned latent tokens. These approaches fix either patch groupings or output token counts; our method instead sets a compression target and learns variable length groups whose boundaries adapt to image content. A more recent work closely related to us is LearnPruner (Takezoe et al., 2026), which trains a learnable pruning module to removes redundant vision tokens. However, unlike VLVLM, it does not train the LLM directly, so the training–inference mismatch remains unresolved.

Multiple works (Chen et al., 2024; Liu et al., 2024b; Wen et al., 2025; Xing et al., 2025; Ye et al., 2025; Huang et al., 2025) also propose to compress image tokens within the LLM layers, leaving early layers processing longer sequences and resulting in less efficiency gains. Our VLVLM compresses tokens before the first LLM layer through a lightweight predictor, with linear cost in the number of patches. This allows *all LLM layers* to process the compressed sequence. Besides, we also show that VLVLM can be combined with these methods (see Table 6 in §5).

Adaptive Tokenization across Modalities. Our work takes inspiration from adaptive tokenization in other modalities such as text and audio. For language modeling, multiple works (Nawrot et al., 2023; Hwang et al., 2025; Pagnoni et al., 2024) propose a boundary predictor to dynamically decide where to merge byte-level representations. Many follow-up works (Owodunni et al., 2026; Ahia et al., 2024) also investigate approaches of learning flexible and adaptive compression rates. To improve efficiency of encoding audio samples, which can be extremely long and contain redundant information and noise, both fixed and adaptive compression techniques have been explored (Zhang et al., 2025a; Zheng et al., 2025; Libera et al., 2026). We adopt these ideas to multimodal settings with the following two key modifications. First, we apply the boundary predictor *after* the entire vision encoder. Second, during inference, we select a subset of boundaries based on their probabilities allowing us to select different compression rates as desired (see Figure 6).

7 Conclusion

We propose VLVLM, a post-training approach to train efficient VLMs, that dynamically chunks image tokens using a lightweight boundary predictor before feeding them to the LLM. Empirical evaluations, with multiple vision encoders as well as LLM backbones and sizes, show that our method outperforms strong training-free and training-based approaches at multiple compression rates, especially on visual tasks which require preserving finer-grained details such text. The simplicity of our approach makes it easy to integrate into existing VLM architectures, motivating future work on larger models, higher-resolution images, and video understanding.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on research methodology or experiments, supporting qualitative and thematic data analysis. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, implementing methods, proposing or refining hypotheses, assisting with translation and interpreting results; Generating synthetic data sets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, cleaning and reformatting dataset are not applicable to this work. Additionally, we used generative AI tools for brainstorming, sourcing/searching for information. We have reviewed all AI-assisted work. We carefully reviewed LLM-generated code for both core implementation of our method and for mundane tasks (e.g., analyze results and draw result figures) and ensured we have verified and understood every step we are doing. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We will open source our code along with model checkpoints and provide clear instructions on how to reproduce our training procedure and follow our evaluation protocol.

REFERENCES

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. Vqa: Visual question answering, 2016. URL https://arxiv.org/abs/1505.00468.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A. Smith. Magnet: Improving the multilingual fairness of language models with adaptive gradient-based tokenization, 2024. URL https://arxiv.org/abs/2407.08818.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models, 2025. URL https://arxiv.org/abs/2503.02175.

Kazi Hasan Ibn Arif, JinYi Yoon, Dimitrios S. Nikolopoulos, Hans Vandierendonck, Deepu John, and Bo Ji. Hired: Attention-guided token dropping for efficient inference of high-resolution vision-language models, 2024. URL https://arxiv.org/abs/2408.10945.

Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection, 2019. URL https://arxiv.org/abs/1904.01941.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

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

Kaitong Cai, Jusheng Zhang, Jing Yang, Yijia Fan, Pengtao Xie, Jian Wang, and Keze Wang. Flashvlm: Text-guided visual token selection for large multimodal models, 2025. URL https://arxiv.org/abs/2512.20561.

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, 2024. URL https://arxiv.org/abs/2403.06764.

Christopher Chou, Lisa Dunlap, Koki Mashita, Krishna Mandal, Trevor Darrell, Ion Stoica, Joseph E Gonzalez, and Wei-Lin Chiang. Visionarena: 230k real world user-vlm conversations with preference labels. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3877–3887. IEEE, 2025.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022a. URL https://arxiv.org/abs/2205.14135.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022b. URL https://arxiv.org/abs/2205.14135.

Jinhong Deng, Wen Li, Joey Tianyi Zhou, and Yang He. Scope: Saliency-coverage oriented token pruning for efficient multimodel llms, 2025. URL https://arxiv.org/abs/2510.24214.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Facebook. fvcore: A lightweight computer vision library, 2019. URL https://github.com/facebookresearch/fvcore.

Zhengyao Fang, Pengyuan Lyu, Chengquan Zhang, Guangming Lu, Jun Yu, and Wenjie Pei. Prune redundancy, preserve essence: Vision token compression in vlms via synergistic importance-diversity, 2026. URL https://arxiv.org/abs/2603.09480.

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, 2025a. URL https://arxiv.org/abs/2306.13394.

Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, Zhang Li, Guozhi Tang, Bin Shan, Chunhui Lin, Qi Liu, Binghong Wu, Hao Feng, Hao Liu, Can Huang, Jingqun Tang, Wei Chen, Lianwen Jin, Yuliang Liu, and Xiang Bai. Ocrbench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning, 2025b. URL https://arxiv.org/abs/2501.00321.

Wenyi Gong and Mieszko Lis. Cubistmerge: Spatial-preserving token merging for diverse vit backbones, 2026. URL https://arxiv.org/abs/2509.21764.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon,

Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam,

Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. URL https://arxiv.org/abs/2111.06377.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Lianyu Hu, Liqing Gao, Fanhua Shang, Liang Wan, and Wei Feng. illava: An image is worth fewer than 1/3 input tokens in large multimodal models, 2026. URL https://arxiv.org/abs/2412.06263.

Wenxuan Huang, Zijie Zhai, Yunhang Shen, Shaosheng Cao, Fei Zhao, Xiangfeng Xu, Zheyu Ye, Yao Hu, and Shaohui Lin. Dynamic-llava: Efficient multimodal large language models via dynamic vision-language context sparsification, 2025. URL https://arxiv.org/abs/2412.00876.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling, 2025. URL https://arxiv.org/abs/2507.07955.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention, 2021. URL https://arxiv.org/abs/2103.03206.

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. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i4.32427. URL https://doi.org/10.1609/aaai.v39i4.32427.

Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i. Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazàn Almazàn, and Lluís Pere de las Heras. Icdar 2013 robust reading competition. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ICDAR '13, pp. 1484–1493, USA, 2013. IEEE Computer Society. ISBN 9780769549996. doi: 10.1109/ICDAR.2013.221. URL https://doi.org/10.1109/ICDAR.2013.221.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023a. URL https://arxiv.org/abs/2301.12597.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models, 2023b. URL https://arxiv.org/abs/2305.10355.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations, 2022. URL https://arxiv.org/abs/2202.07800.

Luca Della Libera, Cem Subakan, and Mirco Ravanelli. Beyond fixed frames: Dynamic character-aligned speech tokenization, 2026. URL https://arxiv.org/abs/2601.23174.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2024a. URL https://arxiv.org/abs/2310.03744.

Ting Liu, Liangtao Shi, Richang Hong, Yue Hu, Quanjun Yin, and Linfeng Zhang. Multi-stage vision token dropping: Towards efficient multimodal large language model, 2024b. URL https://arxiv.org/abs/2411.10803.

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?, 2024c. URL https://arxiv.org/abs/2307.06281.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12), December 2024d. ISSN 1869-1919. doi: 10.1007/s11432-024-4235-6. URL http://dx.doi.org/10.1007/s11432-024-4235-6.

Pan Lu, Swaroop Mishra, Tony 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, 2022. URL https://arxiv.org/abs/2209.09513.

Qiankun Ma, Ziyao Zhang, Haofei Wang, Zhen Song, Jie Chen, and Hairong Zheng. Apet: Approximation-error guided token compression for efficient vlms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26306–26316, June 2026.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models, 2025. URL https://arxiv.org/abs/2504.05299.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, Megh Thakkar, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. Chartqapro: A more diverse and challenging benchmark for chart question answering, 2025. URL https://arxiv.org/abs/2504.05506.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for vqa on document images, 2021. URL https://arxiv.org/abs/2007.00398.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. Efficient transformers with dynamic token pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6403–6417. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.353. URL http://dx.doi.org/10.18653/v1/2023.acl-long.353.

Abraham Toluwase Owodunni, Orevaoghene Ahia, and Sachin Kumar. Flexitokens: Flexible tokenization for evolving language models, 2026. URL https://arxiv.org/abs/2507.12720.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. Byte latent transformer: Patches scale better than tokens, 2024. URL https://arxiv.org/abs/2412.09871.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models, 2026. URL https://arxiv.org/abs/2403.15388.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read, 2019. URL https://arxiv.org/abs/1904.08920.

Rinyoichi Takezoe, Yaqian Li, Zihao Bo, Anzhou Hou, Mo Guang, and Kaiwen Long. Learnpruner: Rethinking attention-based token pruning in vision language models, 2026. URL https://arxiv.org/abs/2604.23950.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features, 2025. URL https://arxiv.org/abs/2502.14786.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025a. URL https://arxiv.org/abs/2508.18265.

Xiangchen Wang, Jinrui Zhang, Teng Wang, Haigang Zhang, and Feng Zheng. Seeing more, saying more: Lightweight language experts are dynamic video token compressors. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings*

of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 541–558, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.28. URL https://aclanthology.org/2025.emnlp-main.28/.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for important tokens in multimodal language models: Duplication matters more, 2025. URL https://arxiv.org/abs/2502.11494.

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, 2025. URL https://arxiv.org/abs/2410.17247.

Tong Xu, Hailong Shi, and Xingyu Gao. Score: Salience-coverage reduction for vision token pruning in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24686–24695, June 2026.

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, 2026. URL https://arxiv.org/abs/2412.04467.

Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. Atp-llava: Adaptive token pruning for large vision language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24972–24982, 2025. doi: 10.1109/CVPR52734.2025.02325.

Hanxun Yu, Wentong Li, Xuan Qu, Song Wang, Junbo Chen, and Jianke Zhu. Visiontrim: Unified vision token compression for training-free mllm acceleration, 2026. URL https://arxiv.org/abs/2601.22674.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities, 2024. URL https://arxiv.org/abs/2308.02490.

Hanglei Zhang, Yiwei Guo, Zhihan Li, Xiang Hao, Xie Chen, and Kai Yu. Unlocking temporal flexibility: Neural speech codec with variable frame rate, 2025a. URL https://arxiv.org/abs/2505.16845.

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, 2025b. URL https://arxiv.org/abs/2412.01818.

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, 2025c. URL https://arxiv.org/abs/2506.10967.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. Sparsevlm: Visual token sparsification for efficient vision-language model inference, 2025d. URL https://arxiv.org/abs/2410.04417.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Rui-Chen Zheng, Wenrui Liu, Hui-Peng Du, Qinglin Zhang, Chong Deng, Qian Chen, Wen Wang, Yang Ai, and Zhen-Hua Ling. Say more with less: Variable-frame-rate speech tokenization via adaptive clustering and implicit duration coding, 2025. URL https://arxiv.org/abs/2509.04685.

Qihui Zhu, Tao Zhang, Yuchen Wang, Zijian Wen, Mengjie Zhang, Shuangwu Chen, Xiaobin Tan, Jian Yang, Yang Liu, Zhenhua Dong, Xianzhi Yu, and Yinfei Pan. Hawk: Head importance-aware visual token pruning in multimodal models, 2026. URL https://arxiv.org/abs/2604.07812.

APPENDIX

The appendix section is structured as follows. Appendix A presents the implementation details of our experiments, including hyperparameter details and what computing environments have been used. Appendix B shows the core algorithmic design (i.e., token downsampling, training and inference of the boundary predictor) of our approach in the form of pseudocode. Appendix C demonstrates detailed efficiency analysis including FLOP-accuracy tradeoff and reduced memory usage. Appendix D provides coarse-grained benchmark-level results. Appendix E illustrates a variety of alternative boundary evaluation and common failure modes. Additional result analysis are presented in Appendix F, such as the compressibility of VQA benchmarks, training dynamics, and the effect of LLM scaling. Appendix G introduces all used VQA benchmarks in more details. We conclude our Appendix with a limitation section Appendix H.

A IMPLEMENTATION DETAILS

Hyperparameters. The sampling temperature for the boundary predictor is set to 0.01 for $4\times$ compression to mitigate noise when a moderate number of image tokens are retained, while it's increased to 1.0 for $8\times$ and $10\times$ compression to help explore boundaries more effectively under a more aggressive compression budget. As noted, our finetuning setup largely mirrors the training recipe of LLaVA 1.5. In the initial token-alignment/pretraining phase, where we train only the MLP projectors (plus the boundary predictor in our method), we adopt DeepSpeed ZeRO-2 with an initial learning rate of 1e-3 and a cosine schedule. The global batch size is 256, and we apply gradient accumulation over 32 steps to reduce memory usage. This pretraining phase is run for exactly 1 epoch. In the subsequent supervised instruction finetuning phase, where the LLM itself is updated (using either full-parameter finetuning or LoRA), we switch to DeepSpeed ZeRO-3 with an initial learning rate of 2e-5 and a cosine scheduler. Here, we use 1 batch per device and accumulate gradients over 128 steps, resulting in a global batch size of 128. This finetuning phase also runs for exactly 1 epoch. In both stages, we set the learning rate warmup ratio to 0.03.

Computing Environments. We run inference on a single NVIDIA A100-40G GPU. The pretraining is likewise carried out on one A100-40G GPU. For finetuning, we use 4 A100-80G GPUs. We use flash attention (Dao et al., 2022a) for all of our training and evaluation experiments.

Synthetic chessboard dataset. All 336×336 synthetic images are partitioned into a 12×12 black-white checkerboards. We divide each image into four quadrants; then we randomly select one cell within each quadrant to put a randomly sampled character there with contrasting color (e.g., if the cell is black, the character will be white, and vice versa). For each image, we construct exactly four questions asking the model to identify the character in each quadrant, asking: what is the character shown on top-left, top-right, bottom-left, bottom-right region, respectively? In total, we generate 200 images and accordingly 800 evaluation prompts.

B ALGORITHM DETAILS

In this section, we present the core algorithmic details and relevant pseudocode regarding our proposed boundary predictor. Recall that for each visual representation z_i at position i, the boundary predictor predicts a boundary probability p_i. During training, we first sample $\hat{p}_i$ from a Bernoulli distribution defined by p_i, which provides a differentiable relaxation of the discrete Bernoulli boundary decisions. Specifically, the sampling step introduces a stochastic random variable $u \sim \mathrm{Uniform}(0, 1)$ as follows:

$$\hat{p}_i = \mathrm{sigmoid}\left[\log \frac{p_i \cdot u}{(1-p_i)\cdot(1-u)}^{1/\tau}\right], \tag{2}$$

where $u \sim \mathrm{Uniform}(0, 1)$ and τ represents a temperature hyperparameter. To obtain the exact boundary decisions $b_i \in \{0, 1\}$, we finally apply thresholding on $\hat{p}_i$ to derive the discrete dynamic boundary sequence: $b_i = \mathbb{I}[\hat{p}_i > 0.5]$. During inference, we select a subset of image tokens with the highest boundary probabilities such that the compression ratio matches the specified compression budget. Algorithm 1 describes how boundaries are determined from image features during training. During inference, we replace the Bernoulli sampling procedure with a `TopK` token selection according

Algorithm 1 Boundary Prediction during Training

Require: Hidden features $\mathbf{H} \in \mathbb{R}^{L \times B \times D}$, sampling temperature τ
Ensure: Soft boundaries $\mathbf{b}^{\text{soft}}$, hard boundaries $\mathbf{b}^{\text{hard}}$

$\mathbf{z} \leftarrow \textsc{BoundaryPredictor}(\mathbf{H})$ ▷ $\mathbf{z} \in \mathbb{R}^{B \times L}$
$\mathbf{p} \leftarrow \sigma(\mathbf{z})$ ▷ $\mathbf{p} \in \mathbb{R}^{B \times L}$
$\mathcal{B} \leftarrow \textsc{RelaxedBernoulli}(\mathbf{p}, \tau)$
$\mathbf{b}^{\text{soft}} \leftarrow \mathcal{B}.\textsc{Sample}()$ ▷ $\mathbf{b}^{\text{soft}} \in \mathbb{R}^{B \times L}$
$\mathbf{b}^{\text{hard}} \leftarrow \mathbb{I}\left[\mathbf{b}^{\text{soft}} > 0.5\right]$ ▷ $\mathbf{b}^{\text{hard}} \in \{0, 1\}^{B \times L}$
return $\mathbf{b}^{\text{soft}}, \mathbf{b}^{\text{hard}}$

to boundary probabilities, as illustrated in Algorithm 2. This modification brings two benefits: first, it guarantees the boundary decisions to be deterministic by removing the sampling procedure; more importantly, `TopK` selection ensures the compression budget at inference time is equal to other baselines we study.

Algorithm 2 Boundary Prediction during Inference

Require: Hidden features $\mathbf{H} \in \mathbb{R}^{L \times B \times D}$, compression ratio ρ
Ensure: Hard boundaries $\mathbf{b}^{\text{hard}}$

$\mathbf{z} \leftarrow \textsc{BoundaryPredictor}(\mathbf{H})$ ▷ $\mathbf{z} \in \mathbb{R}^{B \times L}$
$\mathbf{p} \leftarrow \sigma(\mathbf{z})$ ▷ $\mathbf{p} \in \mathbb{R}^{B \times L}$
$K \leftarrow \text{integer}(L \cdot \rho)$ ▷ calculated based on compression ratio
$\mathbf{b}^{\text{hard}} \leftarrow \text{TopK}(\mathbf{p}, K)$ ▷ exactly K selected tokens
return $\mathbf{b}^{\text{hard}}$

In addition, Algorithm 3 shows how image tokens are being merged given the predicted boundary map. We start with calculating the maximum number of segments in the given batch. Then, we need to derive the segment indices (i.e., to which segment each image token belongs) by computing the cumulative sum from the boundary map. As shown in Algorithm 4, in the cumulative sum operation,

Algorithm 3 Downsample Function for Image Token Merging

Require: Boundaries $\mathbf{b} \in \mathbb{R}^{B \times L}$, hidden features $\mathbf{H} \in \mathbb{R}^{L \times B \times D}$
Ensure: Shortened hidden features $\mathbf{H}' \in \mathbb{R}^{S \times B \times D}$

$S \leftarrow \max\left(\sum_{l=1}^{L} \mathbf{b}_{:,l}\right)$ ▷ maximum number of segments in the batch
$\mathbf{s} \leftarrow \text{detach_cumsum}(\text{right_shift}(\mathbf{b}), \text{dim} = 1)$ ▷ segment indices, $\mathbf{s} \in \mathbb{R}^{B \times L}$
$\mathbf{O}_{b,l,j} \leftarrow \widetilde{\mathbb{I}}[s_{b,l} = j], \quad j = 0, \ldots, S-1$ ▷ **differentiable** one-hot segment assignment
$\mathbf{W} \leftarrow \mathbf{O} \,/\, \sum_{l=1}^{L} \mathbf{O}_{:,l,:}$ ▷ normalized averaging weights
$\mathbf{H}' \leftarrow \text{einsum}(\texttt{"lbd,bls->sbd"}, \mathbf{H}, \mathbf{W})$ ▷ $\mathbf{H} \in \mathbb{R}^{L \times B \times D} \rightarrow \mathbf{H}' \in \mathbb{R}^{S \times B \times D}$
return $\mathbf{H}'$

in order to reach desired boundary prediction from the boundary predictor, we revise it in a way that we prevent gradients from flowing to unwanted elements. Concretely, we ensure that gradients are preserved only for the *current* element by detaching them from all *preceding* elements. Once we obtain the segment indices for all image tokens, we have to perform one-hot assignment (i.e., building a one-hot vector) to properly assign tokens to segments. Mathematically, it should be simply applying the indicator function; however, in a practical setting of training the boundary predictor, this non-differentiable operation may kill useful gradients which could have been received within this downsample process. Therefore, we design another helper function in Algorithm 5, which takes advantage of ReLU functions to make the one-hot operation actually differentiable. With these two helper functions, we end up optimizing the gradients more smoothly and properly for jointly training the boundary predictor. Once the one-hot assignment between tokens and segments is complete, continuing in Algorithm 3, we normalize the weights and finally perform downsampling with a batch-level matrix multiplication.

Algorithm 4 detach_cumsum

Require: Input $\mathbf{x} \in \mathbb{R}^{B \times L}$
Ensure: Cumulative values $\mathbf{c} \in \mathbb{R}^{B \times L}$
1: $\mathbf{x}_{\text{past}} \leftarrow \text{right_shift}(\mathbf{x})$ ▷ collect preceding elements
2: $\mathbf{p} \leftarrow \text{stopgrad}\left(\text{cumsum}(\mathbf{x}_{\text{past}}, \text{dim} = 1)\right)$ ▷ detach gradients from preceding elements
3: $\mathbf{c} \leftarrow \mathbf{x} + \mathbf{p}$ ▷ preserve gradient *only through the current element*
4: **return** $\mathbf{c}$

Algorithm 5 diff_indicator (Differentiable Indicator Function)

Require: Segment index $s_{b,l}$ from s, segment j where $0 \leq j \leq S-1$
Ensure: Differentiable indicator $\widetilde{\mathbb{I}}[s_{b,l} = j]$
1: $\text{box}(x) \leftarrow \text{ReLU}(x) - \text{ReLU}(x-1)$ ▷ differentiable *clamp* from 0 to 1
2: $\widetilde{\mathbb{I}}[s_{b,l} = j] \leftarrow \text{box}(s_{b,l} - j + 1) - \text{box}(s_{b,l} - j)$ ▷ differentiable version of $\mathbb{I}[s_{b,l} = j]$
3: **return** $\widetilde{\mathbb{I}}[s = j]$

C Efficiency Analysis

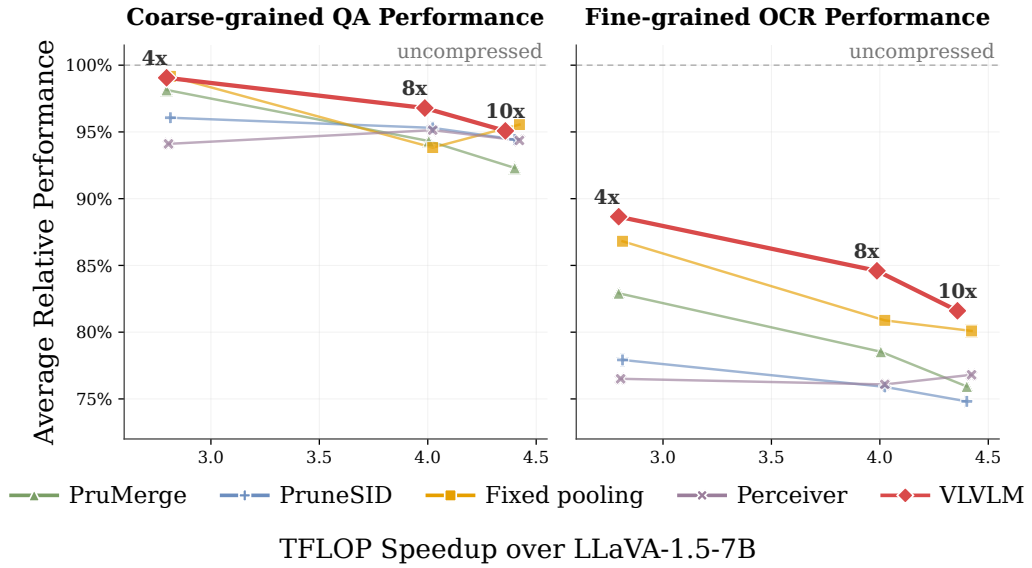


Figure 7: VLVLM delivers the most favorable compute-performance tradeoff.

FLOP-Accuracy Tradeoff. In Figure 7, we examine the tradeoff between relative improvements in TFLOPs and relative retention in task performance. VLVLM reaches the most favorable FLOP-accuracy tradeoff, for both coarse-grained QA and fine-grained OCR performance.

Reduced memory usage. we report memory metrics such as KV cache and allocated memory across different compression budgets in Figure 8. We find that when the number of image tokens is reduced, not only the TFLOP drops, but also the KV cache and allocated memory are less consumed due to the much reduced sequence length. Additionally, as shown in Table 1, boundary predictor leads to only a negligible increase in TFLOPs compared with using fixed pooling. In Table 7, we also report the number of parameters to assess the overhead in terms

Table 7: Boundary Predictor Overhead.

Module	#Params
Vicuna-1.5-7B	6.61B(95.26%)
ViT-L/14	303.51M(4.37%)
MLP projector	20.98M(0.30%)
Boundary predictor	4.20M(**0.06%**)

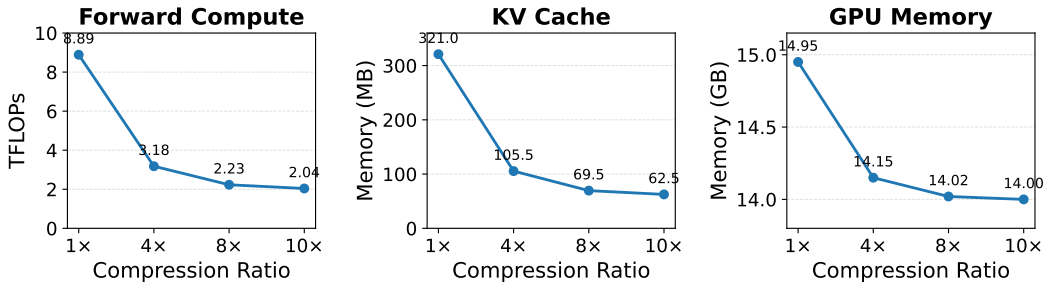


Figure 8: Both KV cache and GPU memory are reduced as compression becomes more aggressive.

of memory. For LLaVA 1.5 model the boundary predictor adds only 0.06% to the total number of parameters.

Compatibility with Flash-attention. Many existing training-free methods such as PruMerge (Shang et al., 2026), iLLaVA (Hu et al., 2026), and Hawk (Zhu et al., 2026) rely on heuristics from attention patterns, making them inflexible and incompatible with efficient attention implementations like Flash-attention (Dao et al., 2022b). Flash-attention provides an IO-aware attention algorithm to reduce the number of memory reads/writes within GPUs. Another advantage of VLVLM is that it is fully independent of the specific implementation of attention and hence Flash-attention can be seamlessly integrated into our training and inference pipeline. Table 8 shows that during the token alignment/pretraining stage of LLaVA, the training efficiency can be largely improved, reaching 1.8× speedup. Our compatibility with Flash-attention allows for much faster training, which otherwise would be impossible if our method were relying on attention heuristics.

Table 8: Compatibility with Flash-attention allows for much faster training.

Attention	pretraining time
w/o Flash-attention	55 hours
w Flash-attention	**30 hours** (1.8×)

D Coarse-grained benchmark-level results

Detailed results on coarse-grained benchmarks. For coarse-grained benchmarks including general QA (VQAv2 (Agrawal et al., 2016), SQA (Lu et al., 2022), MME (Fu et al., 2025a), MM-Bench (Liu et al., 2024c)), reasoning (GQA (Ainslie et al., 2023), MMMU (Hendrycks et al., 2021)), hallucination detection (POPE (Li et al., 2023b)), and free-response QA (LLaVA-B (Liu et al., 2023), MM-Vet (Yu et al., 2024)), we report detailed benchmark-level results for our experiments on LLaVA, SigLIP2, and Qwen2.5 in Table 9, Table 10, Table 11, respectively. We observe that overall, VLVLM preserves up to 99% performance of the original uncompressed model under moderate compression (4×), and maintains 95% performance under more aggressive compression (8× and 10×).

E Additional Boundary Analyses

Failure Mode Analysis. we identify three failure modes of VLVLM for OCR-related questions: scene noise; vertical orientation; ambiguous question. Figure 9 shows that if the scene contains apparent noise around the text region being asked for (e.g., the column is blocking the text), the model's answer may also include such noise and lead to incorrect answer (e.g., "Air-y" instead of "Airy"). Secondly, when the text is positioned vertically rather than horizontally (e.g., the phrase "Cake days" is vertically oriented), due to 1D downsampling, model may not be able to keep track of textual context. Lastly, if the wording of the question itself is ambiguous (e.g., the query of "What *kind of* beer" can be interpreted as either the category or the brand of the beer), the model may produce unintended answer. To address these issues, future work may explore 2-dimensional ways of downsampling the image tokens (Gong & Lis, 2026), as well as principled ways of adaptive compression conditioned on prompts (Yu et al., 2026; Zhang et al., 2025d).

Table 9: Detailed results on LLaVA 1.5 7B on other coarser-grained benchmarks. The best and second best results are highlighted among training-free methods and training-based methods where $*$ indicates statistically significant improvements against fixed pooling.

Model	VQAv2	SQA	MME	MMBench	GQA	MMMU	POPE	LLaVA-B	MM-Vet	Avg. Ret.	TFLOPs
upper bound (keep 576/576 tokens out of the image encoder)											
LLaVA-1.5-7B	76.75	68.42	1425.34	74.43	58.34	34.9	85.18	57.4	29.0	100%	8.89
4× compression (keep 144/576 tokens out of image encoder)											
PruMerge	74.01	68.07	**1408.83**	**73.66**	55.04	34.6	81.36	61.7	27.0	98.14%	3.18(2.8×)
PruneSID	73.44	67.33	1290.23	72.74	54.86	33.1	79.92	59.9	27.6	96.06%	3.16(2.8×)
Perceiver	71.05	67.97	1306.41	71.56	52.50	33.2	81.50	60.5	23.5	94.10%	3.17(2.8×)
fixed pooling	**74.06**	67.82	1335.40	71.72	**55.67**	34.3	**83.75**	**64.1**	**29.9**	**99.16%**	3.16(2.8×)
VLVLM	73.93	**68.72**	1363.67*	73.27*	55.38	**35.6**	83.28	60.6	29.1	99.05%	3.18(2.8×)
8× compression (keep 72/576 tokens out of the image encoder)											
PruMerge	70.67	68.07	1311.21	72.33	52.01	**34.3**	77.53	58.8	25.3	94.31%	2.22(4.0×)
PruneSID	**72.37**	68.67	1280.62	**73.02**	**54.10**	33.8	79.33	57.6	26.7	95.30%	2.21(4.0×)
Perceiver	70.64	69.21	**1379.20**	72.52	52.61	33.4	79.69	**61.0**	24.1	95.13%	2.21(4.0×)
fixed pooling	72.13	69.11	1280.78	71.65	53.78	34.0	82.15	50.7	25.8	93.82%	2.21(4.0×)
VLVLM	72.11	**69.36**	1352.07*	71.35	53.66	33.9	**83.16***	60.0	**27.2**	**96.79%**	2.23(4.0×)
10× compression (keep 58/576 tokens out of the image encoder)											
PruMerge	69.15	68.17	1258.86	71.40	50.90	33.6	76.40	52.5	26.7	92.30%	2.02(4.4×)
PruneSID	**71.67**	68.67	1297.33	**73.13**	**53.17**	33.9	78.76	55.8	25.8	94.43%	2.02(4.4×)
Perceiver	70.70	68.77	1325.25	71.33	52.36	33.6	79.63	**59.6**	24.5	94.36%	2.01(4.4×)
fixed pooling	71.49	68.02	**1359.05**	72.17	53.02	34.3	**82.14**	57.1	26.1	**95.54%**	2.01(4.4×)
VLVLM	70.86	**69.46**	1309.44	70.80	52.76	**34.4**	81.97	55.2	**27.1**	95.07%	2.04(4.4×)

Table 10: Detailed results on LLaVA 1.5 7B with SigLIP2 image encoder on other coarser-grained benchmarks. The best and second best results are highlighted among training-based methods where $*$ indicates statistically significant improvements against fixed pooling.

Model	VQAv2	SQA	MME	MMBench	GQA	MMMU	POPE	LLaVA-B	MM-Vet	Avg. Ret.	TFLOPs
upper bound (keep 576/576 tokens out of the image encoder)											
ViT-L-16-siglip2-384	79.41	71.05	1514.87	76.65	60.38	34.9	87.44	65.5	29.1	100%	8.89
4× compression (keep 144/576 tokens out of image encoder)											
Perceiver	74.58	**72.14**	1384.61	75.03	54.42	33.7	83.46	58.4	28.1	94.73%	3.16(2.8×)
fixed pooling	**77.28**	68.91	**1418.32**	75.51	58.06	**35.3**	**86.39**	**63.2**	**28.3**	**97.36%**	3.16(2.8×)
VLVLM	76.82	69.01	1381.33	**75.71**	**58.67**	34.4	**86.39**	**63.2**	28.0	96.78%	3.18(2.8×)
8× compression (keep 72/576 tokens out of the image encoder)											
Perceiver	73.82	**71.94**	**1384.35**	**74.69**	53.51	34.3	81.31	62.3	27.8	94.84%	2.21(4.0×)
fixed pooling	**75.43**	68.22	1309.92	74.27	**56.44**	**35.1**	**85.36**	61.6	27.7	95.03%	2.21(4.0×)
VLVLM	73.96	69.26	1340.40*	73.73	55.18	34.2	83.10	**64.1**	**30.2**	**95.70%**	2.23(4.0×)
10× compression (keep 57/576 tokens out of the image encoder)											
Perceiver	73.51	**71.74**	1343.66	**75.19**	53.57	34.6	82.50	**64.0**	27.2	94.85%	2.01(4.4×)
fixed pooling	74.96	69.41	1361.73	74.27	**56.29**	33.3	**84.63**	61.7	27.6	94.81%	2.01(4.4×)
VLVLM	**74.98**	69.86	**1376.57**	73.25	55.54	**36.3***	84.19	61.9	**28.4**	**95.95%**	2.04(4.4×)

Reconstruction-oriented boundary evaluation. We also evaluate the predicted boundaries under a different hypothesis: if the boundaries are better under VLVLM, the critical regions within an image should be reconstructed better via an MAE (He et al., 2021) model[5] when other image patches are being masked. Figure 10 shows that under 25% compression ratio, even though fixed pooling and VLVLM reach similar reconstruction loss globally (1.212 vs. 1.207), our method is able to reconstruct the *critical parts of the image* better, namely the display board with much denser information than the grass below.

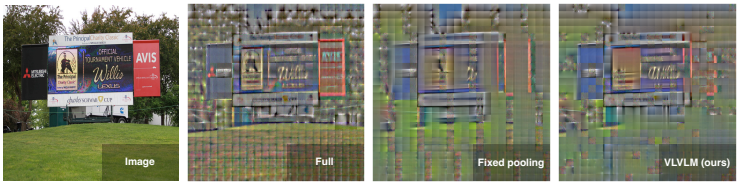


Figure 10: VLVLM reconstructs the fine-grained regions better than fixed pooling.

[5]We use the MAE of H (i.e., huge) size since it has the same patch size (i.e., 14) as CLIP image encoders used in our experiments to avoid resolution mismatch.

Table 11: Detailed results on LLaVA 1.5 with Qwen2.5-14B LLM on other coarser-grained benchmarks. The best and second best results are highlighted among training-free methods and training-based methods where $*$ indicates statistically significant improvements against fixed pooling.

Model	VQAv2	SQA	MME	MMBench	GQA	MMMU	POPE	LLaVA-B	MM-Vet	Avg. Ret.	TFLOPs
upper bound (keep 576/576 tokens out of the image encoder)											
LLaVA Qwen2.5-14B	77.63	76.00	1573.12	79.19	58.92	47.6	85.16	70.2	30.7	100%	17.71
4× compression (keep 144/576 tokens out of image encoder)											
PruMerge	74.06	75.31	1490.82	77.93	54.60	46.4	81.79	66.5	26.0	94.80%	5.62(3.2×)
PruneSID	74.29	74.32	1487.53	78.00	54.86	47.0	83.20	68.7	26.1	95.44%	5.60(3.2×)
Perceiver	70.72	75.56	1482.94	77.31	52.70	47.8	82.45	61.7	23.1	92.47%	5.60(3.2×)
fixed pooling	**75.06**	74.17	**1510.43**	78.16	55.75	48.3	83.96	66.4	27.2	96.31%	5.60(3.2×)
VLVLM	74.90	**76.35**	1497.72	**78.98**	**56.12**	**49.3**	**84.34**	**69.3**	**31.1**	**98.86%**	5.63(3.2×)
8× compression (keep 72/576 tokens out of the image encoder)											
PruMerge	70.89	75.21	1388.75	77.02	52.59	46.9	78.73	65.5	23.9	91.91%	3.61(4.9×)
PruneSID	**73.00**	74.76	1434.29	77.66	**54.38**	**47.9**	83.24	66.3	25.0	94.24%	3.58(4.9×)
Perceiver	68.87	**77.59**	1399.61	76.31	52.20	47.4	80.25	61.8	21.7	90.81%	3.58(4.9×)
fixed pooling	72.71	75.16	1422.09	77.15	54.01	47.6	83.59	64.5	24.5	93.54%	3.58(4.9×)
VLVLM	72.86	76.25	**1440.47**	**77.79**	53.97	46.9	**83.79**	**67.4**	**27.5**	**95.34%**	3.61(4.9×)
10× compression (keep 58/576 tokens out of the image encoder)											
PruMerge	69.38	75.21	1352.54	76.42	51.53	46.9	77.34	64.6	24.5	91.04%	3.18(5.5×)
PruneSID	**72.25**	74.62	1441.86	**77.11**	53.67	**48.0**	82.88	**67.3**	**24.9**	94.05%	3.16(5.5×)
Perceiver	55.56	73.67	1092.56	56.71	45.36	45.8	74.95	28.1	14.9	73.26%	3.16(5.5×)
fixed pooling	72.04	75.31	1427.13	76.47	**54.27**	47.6	**83.95**	65.6	23.1	93.17%	3.16(5.5×)
VLVLM	71.26	**75.61**	**1488.41**	75.37	53.63	46.7	83.86	66.4	**27.0**	**94.57%**	3.22(5.5×)

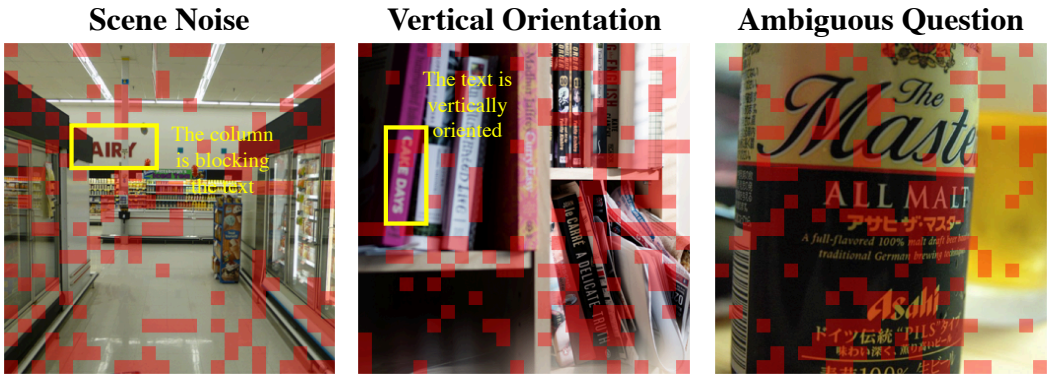


Figure 9: Failure mode analysis. VLVLM is still sensitive to scene noise, vertical oriented text, and ambiguous/vague questions.

CLS token-based boundary evaluation. To further assess the effectiveness of VLVLM, we compare our predicted boundaries against the attention maps of the CLS token. The attention received by the CLS token, commonly used as a simple proxy for token importance (Liang et al., 2022), indicates which image tokens encode the most salient semantics of an image. Using the CLS token attention averaged across all heads, Figure 11 shows that our selected boundaries largely encompass the image tokens that obtain the highest CLS-token attention.

F Additional Result Analysis

Benchmark Compressibility. We further study the ***compressibility*** of each VQA benchmark by examining their relative performance under token compression. Figure 12 presents all 14 VQA benchmarks ranked by performance retention. We observe that benchmarks such as LLaVA-in-the-wild, SQA, MMMU, and MM-Bench are highly compressible, suggesting that they primarily

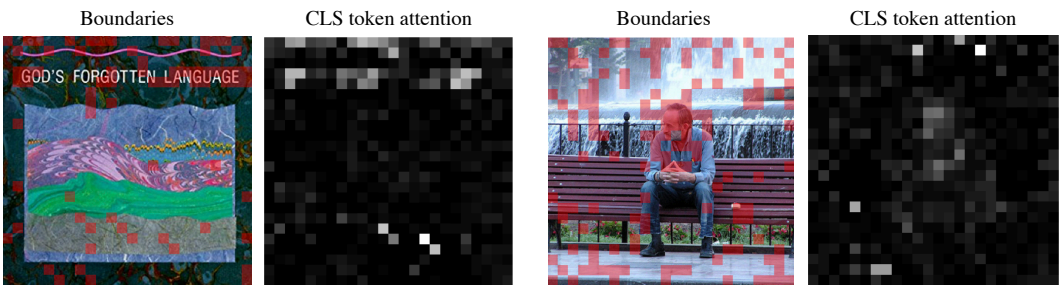


Figure 11: Qualitative validation against CLS token attention maps. Our selected boundaries largely encompass the image tokens that obtain the highest CLS-token attention.

depend on a global understanding of the visual input. In contrast, most OCR benchmarks, which need fine-grained recognition, are highly sensitive to token compression. Nonetheless, our VLVLM approach is able to recover more accuracy on these OCR tasks.

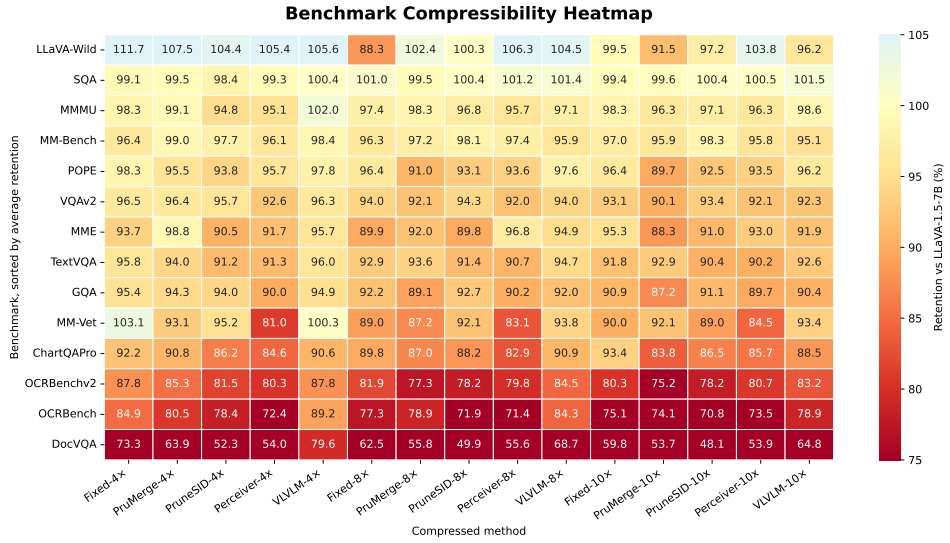


Figure 12: Benchmark Compressibility Heatmap.

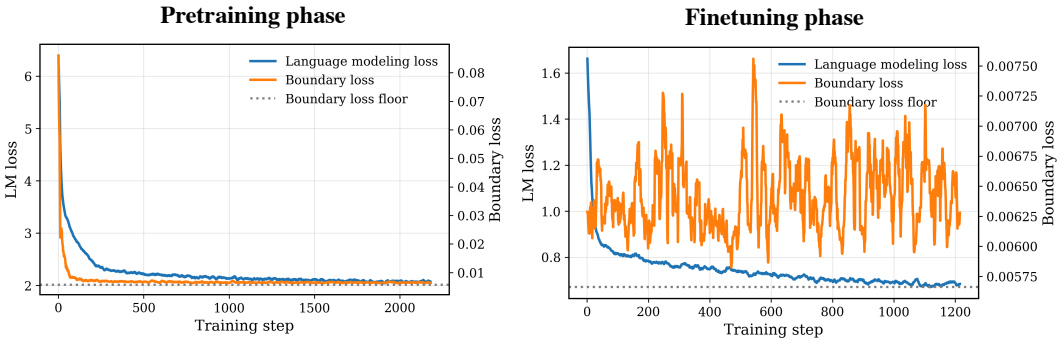


Figure 13: The loss curve for both LM loss and boundary loss in the pretraining phase (left) and finetuning phase (right).

Training dynamics. We additionally present the training loss curves for both the token alignment/pretraining phase and the instruction finetuning phase, as shown in Figure 13. To derive the theoreti-

cal lower bound of the boundary loss, we consider a binomial random variable $K \sim \text{Binomial}(L, \phi)$, where L is the sequence length and ϕ is the compression ratio. Our boundary loss in Equation 1 can then be written as

$$\mathcal{L}_{boundary} = -\frac{1}{L} \log P(K = k). \tag{3}$$

This loss is minimized when $k = L \cdot \phi$. Hence, the theoretical minimum of the boundary loss, which is nonzero, is given by

$$\mathcal{L}_{min} = -\frac{1}{L} \log P(K = L \cdot \phi). \tag{4}$$

In Figure 13, we assume a `4x` compression setting and also plot this theoretical bound for the boundary loss. Concretely, at the beginning of pretraining, both the LM loss and the boundary loss are very large because the boundary predictor is randomly initialized and does not yet enforce the target compression ratio. Both losses then drop rapidly, with the boundary loss converging particularly fast. During instruction finetuning, the LM loss starts relatively high (above 1.6) and decreases gradually over training steps, while the boundary loss, which is already very small after pretraining, fluctuates around the theoretical minimum. It does not exactly attain this minimum because boundary positions are continually sampled during training.

Effect of LLM Scaling. Since we have scaled from Vicuna 7B LLM model to Qwen 2.5 14B, it would be interesting to measure the according performance gain of our proposed method and compare it with baselines such as fixed pooling. We exclusively analyze the performance gain on OCR benchmarks. Figure 14 presents the amount of OCR performance improvement after LLM scaling. It first illustrates that a larger and better LLM (i.e., Qwen 2.5 14B) can bring significant performance gain for both fixed pooling and VLVLM. More importantly, under 4× and 8 × compression, VLVLM even obtain larger scaling benefits than fixed pooling.

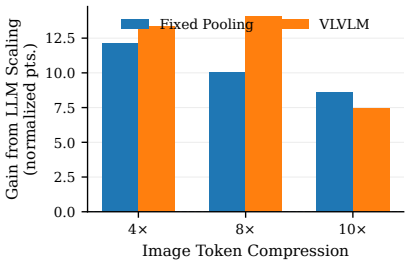


Figure 14: Effect of LLM Scaling.

G VQA BENCHMARKS

In this section, we introduce each of 14 VQA benchmarks we use in our experiments.

VQAv2. VQAv2 (Agrawal et al., 2016) is a large scale VQA benchmark that consist more than 265K open-ended visual questions. All questions are paired with 10 ground truth answer candidates to accurately evaluate the model's predictions. Given the fact the ground truths are not publicly available, all evaluation results will need to be submitted to EvalAI for scoring.

SQA. SQA, also known as ScienceQA (Lu et al., 2022), is a VQA benchmark especially for science domains. Questions are organized hierarchically, covering more than 25 topics. Note that this benchmark contains multimodal questions and text-only questions: to rigorously examine visual understanding capabilities of models, we only report the aggregated accuracy results of multimodal questions.

MME. The MME benchmark (Fu et al., 2025a) evaluates model performance across a total of 14 tasks targeting both perceptual and cognitive abilities. We only report the score of the *perceptual split* since cognitive abilities have little to do with visual compression. MME tries to prevent data leakage for fair evaluation by manually designed instruction-answer pairs.

MMBench. MMBench (Liu et al., 2024c), a bilingual (English & Chinese) VQA multiple-choice benchmark, is rigorously curated with well-designed quality control. It also introduces a rigorous `CircularEval` strategy to ensure the model is robust to the order of choices (i.e., A, B, C, D). We report our results with a single round.

GQA. The GQA (Ainslie et al., 2023) benchmark evaluates visual scene understanding and reasoning capabilities from both images and scene graphs. It also includes spatial attributes and object features paired with well-designed questions. However, we follow LLaVA-1.5 (Liu et al., 2024a) pipeline to exclude object-level questions during our evaluation.

MMMU. MMMU (Hendrycks et al., 2021) evaluates multimodal models on hard tasks that require a wide range of knowledge and strong reasoning abilities. The benchmark consists of more than 10K questions spanning spanning six different disciplines, sourced from exams and textbooks. We only report the overall aggregated accuracy for the simplicity of comparison.

TextVQA. TextVQA (Singh et al., 2019) evaluates a MLLM's ability to capture text embedded within images. It's the most widely used and popular OCR benchmark, which provides a list of candidates for ground truth answers to mitigate the effect of formatting issues in model's responses.

OCRBench. OCRBench (Liu et al., 2024d) is a comprehensive OCR evaluation benchmark. It contains almost 30 datasets spanning different forms of embedded text: multilingual text, handwritten text, non-semantic text, and mathematical expression recognition.

OCRBenchv2. As an extension of OCRBench, OCRBenchv2 (Fu et al., 2025b) presents a larger-scale bilingual text-centric benchmark and provides $4\times$ more tasks than OCRBench. Concretely, it covers more than 30 diverse scenarios with 10K human-verified question-answering pairs.

DocVQA. DocVQA (Mathew et al., 2021) specifically offers a testbed for document images, consisting of 50K questions within 12K document images. We use $Accuracy@NL < 0.5$, which quantifies the percentage of prediction examples whose normalized edit distance is less than 0.5, as our evaluation metric reported in this paper.

ChartQAPro. ChartQAPro(Masry et al., 2025) is a recent benchmark that consists of over 1K chart images such as dashboards and infographics. It's characterized by almost 2K questions spanning multiple-choice, conversational, hypothetical, and unanswerable ones.

POPE. POPE (Li et al., 2023b) evaluates object hallucination in models via curated binary questions (i.e., 'yes' or 'no') that asks whether something is present in the given image. Three splits of hallucinated objects are provided: `random` (a randomly selected object), `common` (a very common object in daily life), and `adversarial` (a very similar but still different object in the image). Metrics like Accuracy, Recall, Precision, and F1 Score measure hallucination levels across three sampling strategies, and we report the arithmetic average of accuracy across 3 different splits above.

LLaVA-in-the-wild. LLaVA-Bench (Liu et al., 2023), also known as LLaVA-in-the-wild, collects a a small but diverse set of 24 images along with 60 questions. All questions expect open-ended free-response answers, thus LLM-as-a-judge is applied to evaluate model's performance.

MM-Vet. MM-Vet (Yu et al., 2024) is a similar free-response benchmark to LLaVA-Bench, but it offers a more structured family of visual questions spanning recognition, OCR, knowledge, language generation, spatial awareness, and math. LLM-as-a-judge is used to evaluate answers from models.

H LIMITATIONS

Due to the high cost of full-parameter finetuning and academic compute constraint, our experiments are limited up to 14B language model. Nevertheless, we provide evidence from Table 3, Table 11, and Figure 14 that 14B model, compared to 7B, consistently brings stronger model performance. Due to the limited availability of existing supervised finetuning data, along with compute constraint, we do not train our models beyond the 665K LLaVA SFT dataset. However, Figure 5 indeed shows the our method has the potential of obtaining better performance with more quality SFT data.