FEWER TOKENS, MORE SELF-TEACHING: ON-POLICY SELF-DISTILLATION FOR EXTREME VISUAL TOKEN REDUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual token reduction is an effective way to accelerate multimodal large language models (MLLMs), but performance deteriorates rapidly under extremely low token budgets. Existing work has explored both visual-token selection and training-based adaptation to reduced visual inputs. We take a step further by asking how a heavily compressed MLLM should learn from the states induced by its own generations. This setting naturally calls for on-policy self-distillation: a heavily compressed model is supervised on the states induced by its own generations, while its full-token counterpart serves as an information-rich teacher. Based on this insight, we propose LT-OPD, a training framework for extreme visual-token reduction. The student rolls out responses with only a small fraction of visual tokens, and a frozen full-token copy of the same MLLM provides distributional supervision along these student-generated trajectories. To stabilize on-policy learning when visual evidence is severely limited, we further introduce a budget-level curriculum that progressively decreases the token budget during training. Across nine benchmarks on Qwen3.5-4B, LT-OPD raises average retained performance under 5% visual-token retention from 68.6% to 82.3%, outperforming training-free, training-based, and reinforcement-learning baselines at the same budget. The gains transfer consistently to Qwen3.5-9B, GLM-4.6V-9B, and LLaVA-OV-1.5-4B. LT-OPD also reduces KV-cache usage by 85.2% and prefill FLOPs by 85.4% without additional inference overhead, demonstrating that on-policy learning can substantially recover capabilities lost to extreme visual-token reduction.

1 INTRODUCTION

Multimodal large language models (MLLMs) (Liu et al., 2023; Hurst et al., 2024; Bai et al., 2025; Team et al., 2025; Wang et al., 2025b; An et al., 2025; Hong et al., 2025; Li et al., 2025) have attracted research interests for years. While exploring the frontiers of MLLMs in many aspects like multimodal understanding and planning, visual question answering and visual agentic planning, the efficiency of such models also matters. When deployed on edge devices or used in resource-constrained scenarios, deploying such MLLMs with billions of parameters remains challenging.

To address this challenge, substantial efforts have been made. Visual token reduction (Chen et al., 2024; Alvar et al., 2025; Zhang et al., 2025b; Wen et al., 2025a; Li et al., 2026a), which reduces the visual tokens from the visual encoder, acts as a training-free and usually plug-and-play strategy in increasing MLLMs' efficiency. However, another problem occurs on applying these training-free techniques. As Figure 1 left suggests, an extremely low visual token budget (ratio of preserved tokens) may be set, like 5%, due to serious resource limits. Under this scenario, focusing on ***which visual tokens should be retained*** cannot achieve acceptable performances of MLLMs.

We argue that, to achieve better performances under extremely low budgets, exploring ***how to adapt a model to operate effectively*** with severely compressed visual inputs also matters. And this needs special adaptive training. Several previous works (Hu et al., 2024; Zhang et al., 2025c; Wen et al., 2025b) support this idea and make efforts on developing training techniques like finetuning with pre-fusion and consistency training. While obvious improvements are seen, a remaining limitation prevents them from achieving further gains. We observe that, low token budgets change not only

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.

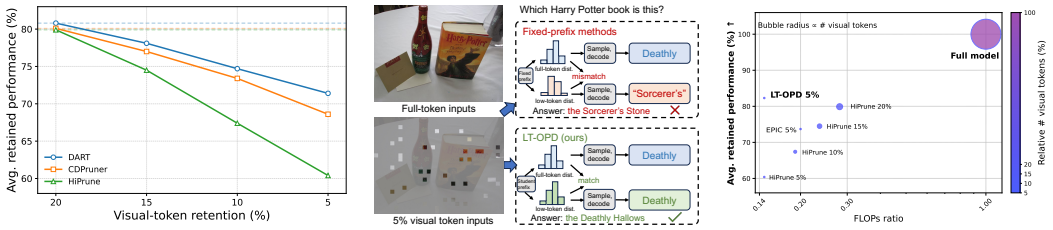


Figure 1: Motivation and performances of LT-OPD. Left: different visual token reduction methods all fail to perform well under an extremely low budget. Middle: Under this scenario, predictive distribution mismatch caused by fixed-prefix methods can lead to the wrong answer. LT-OPD, however, leverages on-policy learning to mitigate the problem. "dist." means distribution and "student prefix" means prefix generated by students. Right: Compared with HiPrune (5%, 10%, 15%, 20%), and EPIC (5%), our LT-OPD reaches the best trade-off.

the amount of visual evidence available, but also the generation trajectories encountered at inference time. As illustrated in Figure 1 middle, with severely limited visual inputs, the predictive distribution of the compressed model can deviate substantially from that of its full-token counterpart. Training directly on fixed response prefixes does not account for the states from the compressed model's own predictions. This mismatch becomes particularly relevant once generated tokens are fed back as subsequent context. This observation naturally motivates us to adapt the compressed model on the trajectories it actually generates, rather than only learning from fixed training prefixes.

Based on this insight, we introduce **L**ow-**T**oken **O**n-**P**olicy **D**istillation (**LT-OPD**), which contains two complementary components. First, we perform a specific on-policy self-distillation algorithm: the low-token student rolls out its own responses. Then a frozen full-token counterpart provides distributional supervision along the prefixes generated by students. In this way, the student learns to recover the behavior of the information-rich teacher exactly at the states it is likely to encounter during inference, as Figure 1 middle shows. Second, directly conducting on-policy learning under an extremely low token budget can be difficult, since the heavily compressed student may struggle to generate useful trajectories at the beginning of training. We therefore introduce a budget-level curriculum. It starts training with a relatively larger budget and progressively reduces it to the target budget. This gradually shifts the student from an easier visual condition to the desired highly compressed, harder condition. This leads to more stable and effective adaptation.

Extensive experiments demonstrate the effectiveness and generality of LT-OPD. Across nine benchmarks, LT-OPD retains 82.3% of the full performance on Qwen3.5-4B while preserving only 5% of visual tokens, substantially improving over the corresponding untrained compressed model (68.6%) and the strongest baseline (77.6%) under the same budget. The improvement consistently transfers across model scales and architectures. Importantly, as Figure 1 right depicts, these gains come without decreasing inference efficiency: the same 0.146× FLOPs ratio as HiPrune (Liu et al., 2026) while improving retained performance by 21.9 percentage points, and 8.6 points higher with 27.0% fewer FLOPs than EPIC (Wen et al., 2025b). In sum, our contributions can be summarized as:

- We revisit extreme visual-token reduction from a training perspective. We highlight that severe visual compression not only removes visual evidence, but also changes the generation states encountered by the compressed model, motivating adaptation directly on its own generated trajectories.
- We propose **LT-OPD**, an on-policy self-distillation framework under this scenario. LT-OPD lets the compressed student learn from a full-token self-teacher on states induced by student, and further introduces a budget-level curriculum that progressively reduces the budget to stabilize learning.
- We conduct extensive evaluations across nine benchmarks, multiple model scales, and three MLLM families. With only **5%** visual tokens, LT-OPD outperforms a wide range of baselines.

2 Related Works

2.1 Token Reduction for Efficient MLLMs

Token reduction is a widely used strategy in enhancing MLLM efficiency. It reduces redundant visual tokens, and therefore mainly reduces prefill time of MLLM inference. Two categories of token reduction methods are developed: training-free methods and training-based ones. For training-free

Table 1: Comparison of representative training-based visual-token reduction methods. "Fixed prefix" means that generation states are determined by static training responses rather than rollouts of the current compressed model.

Method	Training Strategy	Prefix / State Source	On-policy
MQT-LLaVA	Randomly samples the number of query tokens and optimizes the language-modeling objective	Fixed training	No
LLaVA-Mini	Trains compression and modality pre-fusion modules through end-to-end instruction tuning	Fixed training	No
EPIC	Progressive token/layer consistency distillation with SFT and KL supervision	Fixed training	No
LearnPruner	Freezes the base VLM and trains the pruning module with next-token prediction and pruning losses	Fixed training	No
LT-OPD	**Rolls out a low-token student and distills a frozen full-token self-teacher with a budget curriculum**	**Student-generated**	**Yes**

methods, FastV (Chen et al., 2024) first utilizes attention scores to determine redundant tokens. Methods like DivPrune (Alvar et al., 2025), DART (Wen et al., 2025a), CDPruner (Zhang et al., 2025b), HiPrune (Liu et al., 2026) and G^{2}TR Li et al. (2026a) further integrate diversity, duplication and so on to judge redundant tokens during reduction. Besides token selection, training-based methods concentrate on teaching the MLLMs with fewer visual tokens how to generate responses similar to those with full visual information. MQT-LlaVA (Hu et al., 2024), LlaVA-Mini (Zhang et al., 2025c), EPIC (Wen et al., 2025b) are representative methods. LearnPruner (Takezoe et al., 2026) learns a task-aware pruning module with next-token prediction supervision. These methods, while achieving notable success, mainly focus on choosing which visual tokens, or training with fixed prefix, as Table 1 depicts. This may stop them from further improvement under our scenario.

2.2 On-policy Distillation

On-policy distillation (OPD) lets the student model optimize on its self-generated trajectories, with supervision from teacher model by token-level KL divergence or other guidances(Agarwal et al., 2024; Xu et al., 2026; Hou et al., 2026; Li et al., 2026b; Xu et al., 2025). By this method, students can learn better from training distribution than off-policy distillation (Chu et al., 2025). Without online exploration, inference-time prefixes may differ from those seen during training, leading to accumulated errors. Recent works such as OPSD (Zhao et al., 2026) utilizes on-policy distillation on LLMs, with using ground-truth as rewards for optimization. Vision-OPD (Yuan et al., 2026) further provides self-teacher with fine-grained visual information to facilitate with on-policy distillation on MLLMs. In sum, previous OPD methods aim at enhancing the policy's performance through stronger teachers or teacher with more information. Based on this, we explore whether OPD can help train a well-performing student with fewer visual tokens preserved.

3 Methodology

LT-OPD is built on a simple framing: a full-token model has access to richer visual evidence for correcting its compressed counterpart, while the compressed model defines the generation states that will actually be encountered at inference time. We therefore let a low-token student generate its own responses and query a frozen full-token copy of the same MLLM for dense distributional supervision along these student-induced trajectories. Figure 2 gives an overview.

3.1 Problem formulation

As Figure 2 depicts, given an image-question pair $x = (I, q)$, let π_{θ_0} denote the pretrained MLLM and N be the number of visual tokens primarily produced by the visual encoder for I. We instantiate a frozen full-token teacher π_T from θ_0 and a student π_S with trainable parameters θ, initialized from the same parameters. For a keep ratio r, a visual-token reduction operator $\mathcal{R}_r$ retains

$$N_{stu} = \mathrm{round}(r * N) \tag{1}$$

visual tokens for the student, whereas the teacher receives all N tokens. Our optimization goal is to adapt the visual tokens of π_S to a target extreme budget $r = \beta$ (e.g., 5%) while recovering as much of the full-token behavior as possible.

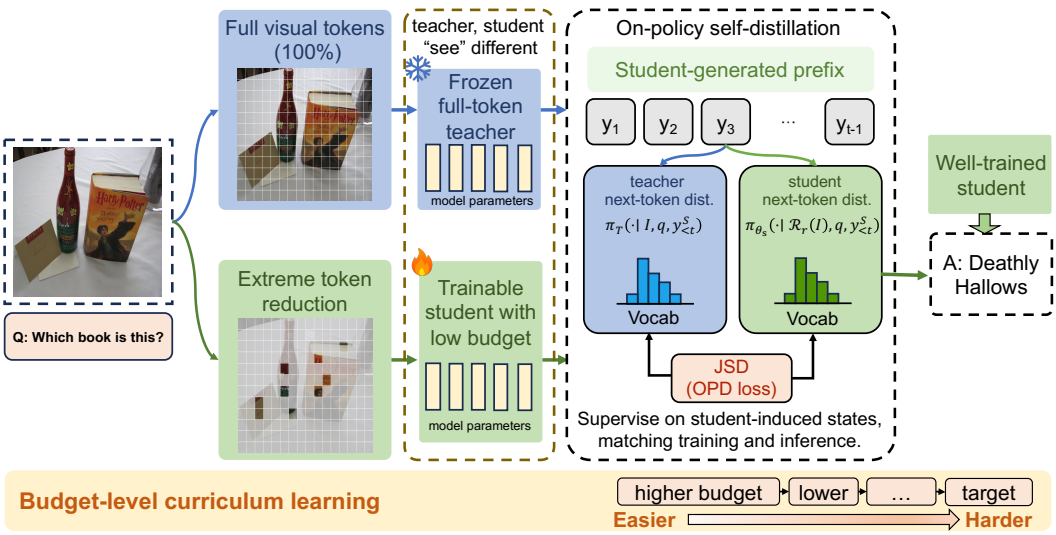


Figure 2: Framework of LT-OPD, consisting of two parts. First, an on-policy self-distillation algorithm is proposed to train a low-token student, with a full-token frozen self-teacher. "dist." means distribution. Second, we design a budget-level curriculum learning process for stable training.

3.2 Our Method: On-policy Self-Distillation across Budgets

Current distillation methods like (Wen et al., 2025b) evaluates the teacher and student on response prefixes provided by a fixed training distribution. Under extreme visual compression, however, the student may quickly deviate from these prefixes, and subsequent predictions are conditioned on states induced by its own previous outputs. LT-OPD instead performs distillation directly on this student-induced state distribution.

Let S be the marker for the "student" and T for the "teacher". At training update s, the low-token student, with parameters θ_s at update s, first samples a response from its current policy,

$$y^S = (y_1^S, \ldots, y_T^S) \sim \pi_{\theta_s}(\cdot \mid \mathcal{R}_r(I), q). \tag{2}$$

The sampled sequence is treated as a stop-gradient trajectory, i.e., we do not differentiate through the discrete sampling process. At each decoding step t, the teacher and student are evaluated on the same student-generated prefix $y_{<t}^S$, but with different amounts of visual evidence:

$$p_{t,\theta}^S = \pi_{\theta_s}\big(\cdot \mid \mathcal{R}_r(I), q, y_{<t}^S\big), \tag{3}$$

$$p_t^T = \pi_T\big(\cdot \mid I, q, y_{<t}^S\big). \tag{4}$$

We minimize the discrepancy between these two next-token distributions π_{θ_s} and π_T. Following prior works (Hou et al., 2026; Yuan et al., 2026), we use the Jensen-Shannon divergence (JSD) (Menéndez et al., 1997):

$$m_{t,\theta} = \frac{1}{2}(p_t^T + p_{t,\theta}^S), \tag{5}$$

$$\ell_t^{\mathrm{JSD}}(\theta) = \frac{1}{2} D_{\mathrm{KL}}(p_t^T \| m_{t,\theta}) + \frac{1}{2} D_{\mathrm{KL}}(p_{t,\theta}^S \| m_{t,\theta}). \tag{6}$$

Accordingly, at update s we optimize the following on-policy distillation surrogate:

$$\widetilde{\mathcal{L}}_{\mathrm{OPD}}^{(s)}(\theta; r) = \mathbb{E}_{(I,q)\sim\mathcal{D}} \mathbb{E}_{y^S \sim \mathrm{sg}[\pi_{\theta_s}(\cdot \mid \mathcal{R}_r(I), q)]} \left[\frac{1}{|y^S|} \sum_{t=1}^{|y^S|} \ell_t^{\mathrm{JSD}}(\theta) \right], \tag{7}$$

where $\mathrm{sg}[\cdot]$ indicates that the rollout distribution is treated as fixed during differentiation. Gradients are taken with respect to the student distributions and evaluated at $\theta = \theta_s$; no score-function gradient is propagated through the sampled tokens. The teacher remains frozen throughout training.

Importantly, the full-token teacher does not generate a separate trajectory. Instead, it provides distributional supervision on prefixes generated by the current compressed student. Thus, "on-policy" refers to the source of the training states: supervision is performed on states induced by the current student policy, rather than on fixed response prefixes. After training, only the compressed student is retained for inference. For implementation efficiency, we use the top-K vocabulary support described in Appendix B; objective ablations are provided in Appendix G.

3.3 Budget-level Curriculum Learning

On-policy supervision is useful only at states that the student can reach. At the beginning of training, directly applying an extreme token budget can produce highly degraded rollouts, making the visited states far from the full-token policy. In other words, the lower the token budget is, more difficult the training task becomes. Motivated by previous work (Bengio et al., 2009), we introduce a budget-level curriculum learning strategy, to progressively increase the task difficulty during training. Specifically, let s_{tol} be the total training steps and s still be the current update, we define a three-stage curriculum process divided by s_1 and s_2, as Figure 3 shows (s_1=14, s_2=100 empirically),

$$r(s) = \begin{cases} (2\alpha+1)\beta, & 0 \le s < s_1, \\ \beta + \alpha\beta\left[1+\cos\left(\pi\frac{s-s_1}{s_2-s_1}\right)\right], & s_1 \le s < s_2, \\ \beta, & s_2 \le s \le s_{tol}, \end{cases} \tag{8}$$

where $r(s)$ is the dynamic budget changed together with s and α is a hyper-parameter giving the starting point of the learning. Here α is set to 2 empirically. β is the final keep ratio. In each training step, the number N_{stu} of visual tokens student policy keep is $\mathrm{Round}(r(s) * N)$.

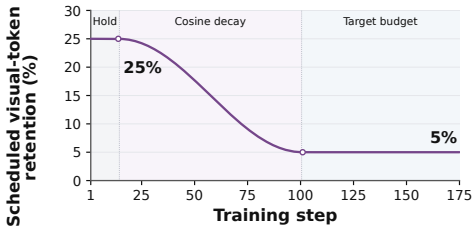


Figure 3: Visual token keep ratio changes.

The first stage provides the student with more visual evidence at a higher budget to establish useful on-policy trajectories. The second stage progressively and smoothly shifts the visited state distribution towards that induced by the target compression level, while the final stage adapts the student directly at the target budget β. The complete LT-OPD objective at update s is therefore simply:

$$\mathcal{L}_{\text{LT-OPD}}(\theta; s) = \widetilde{\mathcal{L}}^{(s)}_{\text{OPD}}(\theta; r(s)). \tag{9}$$

3.4 Training Data Construction

We want the student model can see data from multiple sources, and then the supervision on student-generated trajectories can benefit from data diversity, and improve generalizability of the trained compressed model. Therefore, we specifically construct a training set, LT-14K, from multiple data sources. The training set contains 14,000 single-image questions from sources like LlaVA-Instruct-665K (Liu et al., 2023), OneThinker (Feng et al., 2026), PixMo (Deitke et al., 2025) and Vision-OPD (Yuan et al., 2026) training data. Detailed distribution of the training set is in Appendix A.

4 Experiments

4.1 Experimental Settings.

Model and baselines. We apply our methods on an advanced MLLM, Qwen-3.5-4B. We apply two kinds of baselines. For training-free baselines, we choose previous SOTA (state-of-the-art) methods like FastV (Chen et al., 2024), DivPrune (Alvar et al., 2025), DART (Wen et al., 2025a), CDPruner (Zhang et al., 2025b), HiPrune (Liu et al., 2026) and ZOO-Prune (Kim et al., 2026), and set 20%, 15%, 10% and 5% keeping budget. For training-based ones, we choose vanilla finetuning (same token reduction method as LT-OPD), LlaVA-Mini Zhang et al. (2025c), EPIC Wen et al. (2025b) and LearnPruner (Takezoe et al., 2026), and set the keeping budget of them all to 5%. Here "*LlaVA-Mini*" refers to the all fusion mechanism and training methods in their paper, which

Table 2: Main results of different visual token reduction methods on Qwen3.5-4B. We **bold** the best and underline the second best scores under *5% budget*. Orange lines are our results. "Avg. Retain Ratio" means average retain ratio compared to upper bound.

Method	V*Bench	HR-4K	GQA	MMMU	MMB	MME	POPE	TVQA	OCRB	Avg. Retain Ratio (%)
Full Visual-Token Models (100% Retention)										
Mimo-VL-RL	83.3	73.5	55.9	66.7	84.4	1,814.7	88.6	76.8	866	-
MiniCPM-V-4.5	70.7	69.6	61.0	67.7	84.2	2,360.6	87.9	80.7	890	-
Qwen3-VL-Instruct-8B	84.8	79.6	61.6	69.6	84.5	2,387.4	89.0	81.7	819	-
Qwen3.5-4B (*Upper Bound*)	84.3	84.4	60.7	77.6	86.4	2,300.8	89.3	81.2	850	100%
Training-Free										
Retain 20% Visual Tokens										
FastV (ECCV24)	80.1	72.8	50.0	45.2	76.2	1,729.4	77.7	47.2	468	76.2%
DivPrune (CVPR25)	80.6	76.5	57.4	43.1	78.6	1,938.8	88.7	70.3	579	85.1%
DART (EMNLP25)	67.0	70.0	54.3	40.8	73.8	1,909.8	84.8	73.8	579	80.8%
CDPruner (NeurIPS25)	69.6	71.8	54.5	47.6	82.0	1,949.2	80.0	63.7	463	80.1%
HiPrune (ACL26)	71.7	72.1	55.4	48.7	81.6	1,943.9	81.6	64.2	384	79.9%
ZOO-Prune (CVPR26)	78.5	75.9	58.0	39.6	75.9	2,009.4	89.2	56.0	556	82.1%
Retain 15% Visual Tokens										
FastV (ECCV24)	78.0	72.0	47.1	46.1	73.0	1,568.5	73.9	39.2	393	71.6%
DivPrune (CVPR25)	78.5	76.6	56.1	42.0	78.0	1,889.4	87.3	66.6	542	82.9%
DART (EMNLP25)	62.8	68.0	52.7	41.1	72.2	1,872.8	83.1	70.2	541	78.1%
CDPruner (NeurIPS25)	64.9	70.6	53.2	48.0	80.6	1,845.3	77.6	59.6	419	77.0%
HiPrune (ACL26)	68.1	72.0	52.3	48.4	78.6	1,750.8	78.6	55.0	279	74.5%
ZOO-Prune (CVPR26)	78.0	75.3	56.5	42.9	74.7	2,000.1	88.8	49.8	503	80.4%
Retain 10% Visual Tokens										
FastV (ECCV24)	73.8	72.0	43.7	46.0	69.1	1,452.4	69.3	35.1	305	67.1%
DivPrune (CVPR25)	74.9	74.5	54.6	42.6	77.8	1,873.8	86.0	60.0	496	80.2%
DART (EMNLP25)	59.2	61.9	51.1	41.2	72.1	1,803.8	81.8	63.5	505	74.7%
CDPruner (NeurIPS25)	57.6	69.6	51.6	46.7	78.4	1,742.2	74.3	53.9	415	73.4%
HiPrune (ACL26)	63.9	70.2	49.0	47.6	75.6	1,471.0	72.2	40.3	202	67.4%
ZOO-Prune (CVPR26)	77.5	73.6	55.2	40.1	74.2	1,949.4	87.8	40.1	469	77.2%
Retain 5% Visual Tokens										
FastV (ECCV24)	63.4	66.5	39.8	45.8	61.2	1,193.2	61.9	31.0	187	59.0%
DivPrune (CVPR25)	70.7	**74.4**	54.6	43.7	77.2	1,834.8	86.0	48.6	467	77.6%
DART (EMNLP25)	52.9	57.6	51.1	42.8	71.8	1,771.3	81.7	52.3	477	71.4%
CDPruner (NeurIPS25)	52.9	68.4	50.4	45.7	74.0	1,531.8	70.2	43.2	404	68.6%
HiPrune (ACL26)	58.7	68.3	45.4	47.2	71.2	1,372.9	63.7	25.5	107	60.4%
ZOO-Prune (CVPR26)	70.7	71.3	54.1	38.3	74.1	1,915.4	85.0	26.8	431	72.7%
Training-Based (5% Visual Tokens)										
LLaVA-Mini (ICLR25)	62.8	70.1	47.8	47.2	78.2	1,879.6	78.0	45.2	370	72.9%
EPIC (NeurIPS25)	68.6	72.8	43.1	47.3	77.8	1,781.9	81.1	40.5	470	73.7%
LearnPruner (ICLR26)	66.0	61.4	48.7	47.1	73.3	1,534.9	73.9	35.0	403	68.5%
5% Token + SFT	53.9	66.2	48.8	47.8	77.4	1,741.7	72.5	51.6	330	70.4%
5% Token + curriculum + SFT	61.8	71.8	49.2	47.4	77.7	1,715.2	72.9	53.7	380	73.1%
5% Token + curriculum + fixed-prefix JSD	65.5	69.3	52.3	48.4	77.7	1,839.0	83.5	53.2	520	77.6%
LT-OPD (w/o curriculum)	74.3	72.1	54.7	49.6	**78.9**	1,995.5	**86.5**	**56.3**	524	81.5%
LT-OPD	**74.9**	73.0	**54.8**	**50.0**	**78.9**	**2,077.0**	86.3	56.0	**535**	**82.3%**

we reproduce on Qwen3.5-4B (Qwen Team, 2026). For fair comparison, we all keep 5% budget for them. We also apply another two MLLMs, GLM-4.6V-9B (Hong et al., 2025) and LlaVA-OneVision (OV)-1.5-4B (An et al., 2025) for cross-model evaluation.

Basic settings. Before trainining, we choose CDPruner (Zhang et al., 2025b), a token pruning method, based on conditional diversity, to reduce visual tokens of the model. It is easy for implementation. We train our model and training-based comparing methods on 8×NVIDIA A100 80G GPUs. Evaluations are done on one same GPU. Detailed settings can be found in Appendix B, C.

Datasets and benchmarks. All the training-based methods are trained on the constructed LT-14K for fair comparison. For evaluations, we choose four kinds of benchmarks: (1) perceptual-heavy benchmarks, V*Bench (Wu & Xie, 2024) and HR-Bench 4K (HR-4K) (Wang et al., 2025c), which require fine-grained visual information; (2) general VQA benchmarks, GQA (Hudson & Manning, 2019), MMMU (Yue et al., 2024), MMB (en) (Liu et al., 2024a) and MME (Fu et al., 2025) (3) visual hallucination benchmarks, POPE (Li et al., 2023); (4) OCR benchmarks, TextVQA (TVQA) (Singh et al., 2019) and OCRBench (OCRB) (Liu et al., 2024b). Evaluation prompts are in Appendix D.

Evaluation metrics. We follow the original metrics for judgment of the chosen benchmarks. Notably, we also calculate the average retain ratio compared to upper bound, to intuitively quantify how much of the full-token model's capability is preserved under token reduction: average retain ratio $= \frac{1}{M}\sum_{i=1}^{M}\frac{score_i}{score_i^{\text{full}}} \times 100\%$, where M benchmarks are chosen and $score_i$, $score_i^{\text{full}}$ mean the scores of compressed and full-token model.

Table 3: Different model scales. We apply LT-OPD on 4B and 9B version. "Avg." means average retain ratio compared to upper bound (4B or 9B full), the same as Table 2.

Method	V*Bench	HR-4K	GQA	MMMU	MMB	MME	POPE	TextVQA	OCRB	Avg.
Qwen3.5-4B (5% visual token)	52.9	68.4	50.4	45.7	74.0	1,531.8	70.2	43.2	404	68.6%
+ LT-OPD	**74.9**	**73.0**	**54.8**	**50.0**	**78.9**	**2,077.0**	**86.3**	**56.0**	**535**	**82.3%**
Qwen3.5-9B (5% visual token)	54.5	59.6	49.4	51.7	77.7	1,809.4	82.3	46.6	430	71.1%
+ LT-OPD	**70.7**	**68.1**	**53.6**	**52.9**	**80.5**	**2,147.9**	**86.2**	**61.5**	**539**	**81.4%**

4.2 Main results

Effectiveness of LT-OPD. Table 2 compares LT-OPD with both training-free and training-based visual token reduction methods on Qwen3.5-4B. Under the extreme 5% visual-token budget, LT-OPD achieves an average retain ratio of 82.3%, outperforming the strongest training-free baseline, DivPrune, by 4.7 percentage points (77.6%), and training-based baseline, EPIC by 8.6 points (73.7%). The improvement is broad rather than benchmark-specific: LT-OPD achieves the best result among prior methods keeping 5% visual tokens on **8 out of 9 benchmarks**, including all categories. The gains are particularly clear on V*Bench (74.9 vs. 70.7), MME (2,077.0 vs. 1,915.4), and OCRBench (535 vs. 477), where the latter numbers denote the best competing results under the same budget. Additionally, the chosen visual token reduction strategy for LT-OPD, CDPruner, achieves an average retain ratio of only 68.6% under 5% budget, much lower than LT-OPD. This suggests that LT-OPD does not rely on a "best" token reduction method, but proper on-policy learning actually works.

More notably, with only 5% visual tokens, LT-OPD still outperforms all training-free methods retaining 10% tokens in average retain ratio (82.3% vs. at most 80.2%), and nearly matches the best result obtained with a 15% or even 20% budget. These results demonstrate that with effective training, our method can recover substantially more model capability even under extreme low token budget than relying on token selection alone or applying simple training.

Ablation studies. We ablate the proposed OPD training and curriculum learning strategy, reported in the last 4 lines of Table 2. It is obvious that both components can improve training performance, as curriculum learning increases average retain ratio of simply SFT by 2.7 percentage points, and OPD-only training reaches 81.5% average retain ratio. Based on this, full-version LT-OPD achieves the best performance (82.3%) and clearly improves model performance across all benchmarks. More notably, training with curriculum and JSD with fixed prefix (knowledge distillation) reaches a score of 77.6, still lower than our results, suggesting that on-policy learning really benefits LT-OPD.

We also test different training objectives displayed in Appendix G. While all training objectives show promising performance, our objective, JSD, is better on perception-heavy benchmarks. Forward/Reverse KL, however, achieve better scores on OCR ones. Ablations on curriculum are done in Appendix H. The whole ablation study suggests the effectiveness and stability of our full design.

4.3 Different Model Scales.

To examine whether LT-OPD remains effective across different model scales, we evaluate it on the 4B and 9B variants of Qwen3.5 under the same 5% visual-token budget. As shown in Table 3, LT-OPD consistently improves both model sizes across all evaluated benchmarks. For Qwen3.5-4B, the average retained performance increases from 68.6% to 82.3%, while for Qwen3.5-9B it improves from 71.1% to 81.4%. Notably, although the larger 9B model provides a stronger compressed baseline, LT-OPD still yields substantial gains, particularly on V*Bench, HR-Bench, and so on. These results indicate that the effectiveness of LT-OPD is not tied to a particular model scale. The benefit of our method persists as the underlying MLLM scales from 4B to 9B parameters.

4.4 Cross-Model Performance.

We apply LT-OPD on another two advanced MLLMs, GLM-4.6V-9B and LlaVA-OV-1.5-4B. Table 4 reports the results. LT-OPD consistently improves the performance of both models under the same 5% visual-token budget, increasing the average retained performance from 73.8% to 86.3% on GLM-4.6V-9B and from 71.8% to 84.1% on LLaVA-OV-1.5-4B. The gains are observed across a broad range of benchmarks, with particularly clear improvements like MMBench and MME. These results demonstrate that the effectiveness of LT-OPD is not specific to the Qwen family, but transfers consistently to different MLLM backbones, suggesting that on-policy self-distillation provides a broadly effective training signal for models under extremely limited visual-token budgets.

Table 4: Performance of LT-OPD on different kinds of MLLMs. "Avg." means average retain ratio compared to upper bound (full version of models), the same as Table 2.

Method	V*Bench	HR-4K	GQA	MMMU	MMB	MME	POPE	TextVQA	OCRB	Avg.
Qwen3.5-4B (5% visual token)	52.9	68.4	50.4	45.7	74.0	1,531.8	70.2	43.2	404	68.6%
+ LT-OPD	**74.9**	**73.0**	**54.8**	**50.0**	**78.9**	**2,077.0**	**86.3**	**56.0**	**535**	**82.3%**
GLM-4.6V-9B + (5% visual token)	45.0	60.5	46.8	42.3	69.2	1,646.0	76.3	56.1	346	73.8%
+ LT-OPD (5% visual token)	**51.3**	**63.4**	**53.8**	**48.3**	**83.4**	**2,146.3**	**83.3**	**64.8**	**497**	**86.3%**
LlaVA-OV-1.5-4B + (5% visual token)	64.9	51.9	50.9	47.8	64.6	1,532.1	81.9	38.6	87	71.8%
+ LT-OPD (5% visual token)	64.9	**54.0**	**55.9**	**49.9**	**78.5**	**1,940.5**	**87.4**	**46.8**	**430**	**84.1%**

Table 5: Efficiency of LT-OPD. Δ means comparing with raw Qwen3.5-4B. On the 5% budget, LT-OPD can reduce prefilling cost a lot. Details are in Appendix E.

Method	KV Cache (MB) ↓		FLOPs Ratio ↓		Prefill FLOPs (T) ↓		End-to-End Latency (ms/sample) ↓	
	Value	Δ	Value	Δ	Value	Δ	Value	Speedup ↑
Qwen3.5-4B (full visual token)	26.59	–	1.000	–	6.002	–	1331.510	1.000×
+ HiPrune (5% visual token)	3.93	85.23%	0.146	85.38%	0.878	85.38%	1308.029	1.018×
+ EPIC (5% visual token)	3.93	85.23%	0.200	80.03%	1.198	80.03%	1296.922	1.027×
+ LT-OPD (5% visual token)	3.93	85.23%	0.146	85.38%	0.878	85.38%	1293.691	1.029×

4.5 Efficiency

Inference efficiency. Table 5 summarizes the inference efficiency of LT-OPD (evaluation details in Appendix E). By retaining only 5% visual tokens, LT-OPD reduces the KV cache by 85.2%, and both overall and prefill FLOPs by 85.4%, while introducing no additional inference overhead. It achieves the same computational cost as HiPrune and requires substantially fewer FLOPs than EPIC (0.146 vs. 0.200), demonstrating that the performance gains of LT-OPD do not compromise the efficiency of the compressed model. Additionally, the end-to-end latency improves moderately from 1331.5 to 1293.7 ms (1.029×), consistent with prior observations (Wen et al., 2025b; Zamini & Shukla, 2026) that aggressive visual-token reduction primarily benefits prefill computation, while autoregressive decoding time is dominated by per-token computation and memory access.

Training efficiency. Our training method trains on only 14K training data. The training loop ran for 15.1h on eight GPUs. Considering the study of training time in the original paper Wen et al. (2025b) that 12.2-48 hours are used for training on the same devices (EPIC, LlaVA-Mini, etc.), our training cost is acceptable under the scenario of training-based visual token reduction.

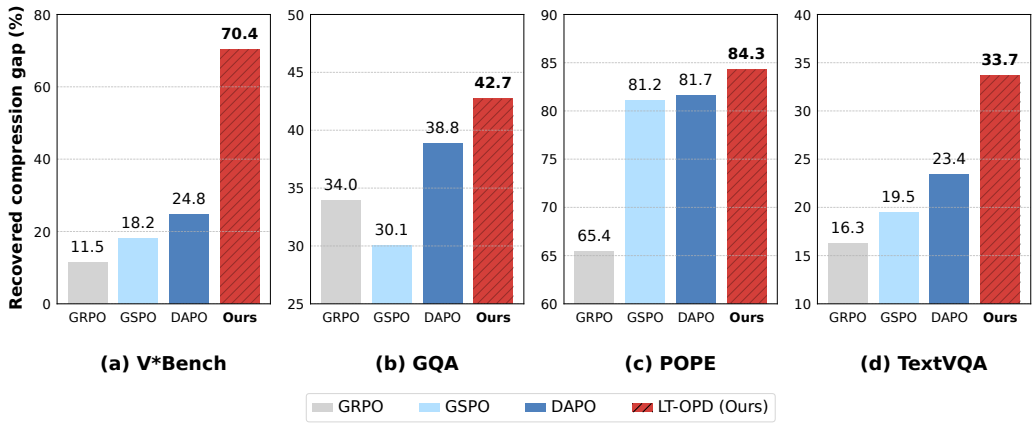


Figure 4: **Recovery at 5% visual-token retention.** We report the scores of LT-OPD and compared RL algorithms. LT-OPD achieves the best recovery performances.

4.6 Compared with RL algorithms

We compare our method with three advanced RL algorithms: GRPO (Shao et al., 2024), GSPO (Zheng et al., 2025) and DAPO (Yu et al., 2025). As shown in Table 6, directly applying these general-purpose RL algorithms substantially improves the 5% visual-token baseline, increasing the average retain ratio from 68.6% to 75.8%–78.2%. Nevertheless, LT-OPD consistently outperforms all three alternatives across all benchmarks and further raises the average retain ratio to

Table 6: Compared with advanced RL algorithms. "Avg." means average retain ratio compared to upper bound, the same as Table 2.

Method	V*Bench	HR-4K	GQA	MMMU	MMB	MME	POPE	TextVQA	OCRB	Avg.
Qwen3.5-4B (5% visual token)	52.9	68.4	50.4	45.7	74.0	1,531.8	70.2	43.2	404	68.6%
+ GRPO	56.5	66.8	53.9	49.0	77.1	1,958.1	82.7	49.4	479	75.8%
+ GSPO	58.6	63.9	53.5	49.9	78.3	2,072.3	85.7	50.6	467	76.8%
+ DAPO	60.7	67.9	54.4	49.6	78.0	2,037.7	85.8	52.1	497	78.2%
+ LT-OPD	**74.9**	**73.0**	**54.8**	**50.0**	**78.9**	**2,077.0**	**86.3**	**56.0**	**535**	**82.3%**

Table 7: Different teacher regularization strategies on Qwen3.5-4B. "Avg." means average retain ratio compared to upper bound, the same as Table 2

Teacher	V*Bench	HR-4K	GQA	MMMU	MMB	MME	POPE	TextVQA	OCRB	Avg.
Initial Policy	74.9	73.0	54.8	50.0	78.9	2,077.0	86.3	56.0	535	82.3%
Trust-Region Regularization	74.3	72.1	52.9	49.0	79.1	2,115.0	86.4	55.4	518	81.5%
EMA	75.4	73.4	54.1	50.0	77.6	2,026.6	84.9	55.7	511	81.3%

82.3%. In particular, compared with the strongest RL baseline DAPO, LT-OPD achieves notable gains on V*Bench, MME, and TextVQA. These results suggest that, under an extremely limited visual-token budget, distilling the model's on-policy knowledge provides a more effective learning signal than optimizing with general-purpose RL objectives, enabling the compressed model to better recover the capabilities of its full-token counterpart.

4.7 Extensive Studies

Training strategies. Table 7 reports performance with different training strategies of OPSD, mentioned in (Zhao et al., 2026; Yuan et al., 2026). Details are in Appendix F. LT-OPD primarily applies the frozen initial policy as the teacher. We discover that LT-OPD keeps stable and good performance across all benchmarks, with the scores and average retain ratio do not change much (81.3%-82.3%). This study supports the robustness of our method even using different regularization methods.

Compression-gap recovery. We want to explore whether our training algorithms better help the student policy to "recover" performance lost to visual token pruning. Thus, we define a new metric, recovery, calculated by $100\,(S_{\text{method}} - S_{\text{pruned}})/(S_{\text{full}} - S_{\text{pruned}})$. $S_{\{\text{full, pruned, method}\}}$ denotes original scores and scores before and after training. Larger recovery represents that a better policy is provided. Figure 4 compares the scores between LT-OPD and RL algorithms. LT-OPD recovers the most across four evaluated benchmarks, especially much higher (over 45 points) than RL methods on V*Bench. This comparison further supports the results in Section 4.6.

Results on multi-image benchmarks. We apply our method and baselines on two multi-image benchmarks, MuirBench (Wang et al., 2025a) and MVBench (Li et al., 2024). Results are in Appendix I. LT-OPD outperforms its training-free counterpart (CDPruner) and EPIC a lot (by over 12% in average retain ratio). We may draw the conclusion that LT-OPD has the potential to be generalized to video understanding tasks, which lead one of the future directions (Zhang et al., 2025d).

Case studies. Some case studies are in Appendix J. These cases range from questions natural images to OCR questions. LT-OPD can do well even on cases needing fine-grained visual information understanding or small texts compared with baselines. This suggests that students trained with LT-OPD have strong potential across different perceptual tasks.

5 Conclusion

Extreme visual-token reduction changes more than the amount of visual evidence available to an MLLM: it also changes the generation states that the compressed model encounters. Motivated by this observation, we introduce LT-OPD, which distills a full-token self-teacher directly on trajectories generated by the low-token student, and progressively reduces the visual-token budget for stable training. Extensive experiments across nine benchmarks, multiple model scales, and different MLLM families show that LT-OPD substantially recovers the capability lost under 5% visual-token retention without sacrificing the efficiency benefits of compression. These results point to a broader direction for efficient MLLMs: under extreme compression, effective token selection and effective adaptation should be considered jointly rather than in isolation.

AI USE STATEMENT

In this work, we used generative AI tools for assist with translation. We have not used generative AI tools for things like generating data and supporting qualitative and thematic data analysis, and the rest of the required disclosure tasks like mathematical proofs are not applicable to this work. Additionally, we used generative AI tools for polishing our language. We have reviewed all AI-assisted work. We checked LLM-generated research ideas for potential plagiarism through a manual literature survey. LLM-generated code was verified and tested for correctness by all of the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our paper does not include any human subjects, practices to data set releases, potentially harmful insights, methodologies and applications, potential conflicts of interest, discrimination/bias/fairness concerns, privacy and security issues, legal compliance, and research integrity issues.

REPRODUCIBILITY STATEMENT

The training dataset details and benchmark details are in Section 3.4, Section 4.1 and Appendix A. Training details are in Appendix B. We also include the prompts in Appendix C and Appendix D. Evaluation protocols on efficiency are in Appendix E. We will release all our codes and useful artifacts once the paper is accepted.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *CVPR*, 2025.

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025.

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, 2024.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *ICML*, 2025.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*, 2025.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *ACM MM*, 2024.

Kaituo Feng, Manyuan Zhang, Hongyu Li, Kaixuan Fan, Shuang Chen, Yilei Jiang, Dian Zheng, Peiwen Sun, Yiyuan Zhang, Haoze Sun, et al. Onethinker: All-in-one reasoning model for image and video. In *CVPR*, 2026.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. In *NeurIPS Track on Datasets and Benchmarks*, 2025.

Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. Glm-4.5 v and glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

Wenjin Hou, Shangpin Peng, Weinong Wang, Zheng Ruan, Yue Zhang, Zhenglin Zhou, Mingqi Gao, Yifei Chen, Kaiqi Wang, Hongming Yang, et al. Uni-opd: Unifying on-policy distillation with a dual-perspective recipe. *arXiv preprint arXiv:2605.03677*, 2026.

Wenbo Hu, Zi-Yi Dou, Liunian H Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka query transformer for large vision-language models. In *NeurIPS*, 2024.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Youngeun Kim, Youjia Zhang, Huiling Liu, Aecheon Jung, Sunwoo Lee, and Sungeun Hong. Zooprune: training-free token pruning via zeroth-order gradient estimation in vision-language models. In *CVPR*, 2026.

Junxian Li, Di Zhang, Xunzhi Wang, Zeying Hao, Jingdi Lei, Qian Tan, Cai Zhou, Wei Liu, Yaotian Yang, Xinrui Xiong, et al. Chemvlm: Exploring the power of multimodal large language models in chemistry area. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 415–423, 2025.

Junxian Li, Kai Liu, Zizhong Ding, Zhixin Wang, Zhikai Chen, Renjing Pei, and Yulun Zhang. G^2tr: Generation-guided visual token reduction for separate-encoder unified multimodal models. *arXiv preprint arXiv:2605.12309*, 2026a.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *CVPR*, 2024.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026b.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *EMNLP*, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

Jizhihui Liu, Guangdao Zhu, Feiyi Du, Niu Lian, Jun Li, Bin Chen, Weili Guan, and Yaowei Wang. Hiprune: Hierarchical attention for efficient token pruning in vision-language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, 2024a.

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*, 2024b.

María Luisa Menéndez, Julio Angel Pardo, Leandro Pardo, and María del C Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 1997.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019.

Rinyoichi Takezoe, Yaqian Li, Zi-Hao Bo, Anzhou Hou, Mo Guang, and Kaiwen Long. Learn-pruner: Rethinking attention-based token pruning in vision language models. In *ICLR*, 2026.

Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. Kimi-vl technical report. *arXiv preprint arXiv:2504.07491*, 2025.

Fei Wang, Xingyu Fu, James Y Huang, Zekun Li, Qin Liu, Xiaogeng Liu, Mingyu Derek Ma, Nan Xu, Wenxuan Zhou, Kai Zhang, et al. Muirbench: A comprehensive benchmark for robust multi-image understanding. In *ICLR*, 2025a.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *AAAI*, 2025c.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *EMNLP*, 2025a.

Zichen Wen, Shaobo Wang, Yufa Zhou, Junyuan Zhang, Qintong Zhang, Yifeng Gao, Zhaorun Chen, Bin Wang, Weijia Li, Conghui He, et al. Efficient multi-modal large language models via progressive consistency distillation. In *NeurIPS*, 2025b.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *CVPR*, 2024.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Wenda Xu, Rujun Han, Zifeng Wang, Long Le, Dhruv Madeka, Lei Li, William Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *ICLR*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *NeurIPS*, 2025.

Qianhao Yuan, Jie Lou, Xing Yu, Hongyu Lin, Le Sun, Xianpei Han, and Yaojie Lu. Vision-opd: Learning to see fine details for multimodal llms via on-policy self-distillation. *arXiv preprint arXiv:2605.18740*, 2026.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *CVPR*, 2024.

Mohamad Zamini and Diksha Shukla. Delta-llava: Base-then-specialize alignment for token-efficient vision-language models. In *WACV*, 2026.

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

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

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

Xiaoyi Zhang, Zhaoyang Jia, Zongyu Guo, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Deep video discovery: Agentic search with tool use for long-form video understanding. In *NeurIPS*, 2025d.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *ICML*, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

A Training Data Details

Training mixture. The distribution of LT-14K are in Table 8. A deterministic balanced sampler realizes these quotas over 175 batches of 80 questions. The Vision-OPD source uses sampling seed 20260830; the remaining sources use 20260824. We also carefully check the training data, and confirm that no data leakage happens.

Table 8: Composition of the 14,000-question training set.

Source / task bucket	Questions
OneThinker: multiple choice	3,000
OneThinker: mathematics	1,500
OneThinker: numerical answer	900
OneThinker: OCR	900
OneThinker: regression	900
PixMo Ask Model Anything	3,000
LLaVA v1.5 mixture	2,800
Vision-OPD full-image multiple choice	1,000
Total	14,000

B Implementation and Training Details

Student and teacher. We initialize the student and teacher from Qwen3.5-4B. The teacher is a frozen copy of the initial model and runs in evaluation mode without gradients. Both receive the same clean full image and question, with the teacher using the dense visual representation. We train the student language model, vision encoder, and native visual merger with full-parameter updates.

Visual token selection. We adapt CDPruner to Qwen3.5 and apply it after the native visual merger and before the language model. The parameter-free selector combines question-conditioned relevance with visual diversity through a matrix-free greedy conditional determinantal-point-process procedure in FP32. Selection runs without gradients, breaks ties by original token index, and preserves the selected embeddings' original multimodal rotary positions.

Distillation objective. Training uses student-generated trajectories and distributional on-policy distillation. We use a Jensen-Shannon objective with mixture coefficient 0.5. At each response position, the support is the union of the student's top 100 tokens, the teacher's top 100 tokens, and the sampled token, with the remaining probability mass represented by one aggregated tail category, following (Yuan et al., 2026). Loss reduction averages valid response-token losses within each sequence and then averages over sequences.

Table 9: Training configuration of the 14K curriculum model. Learning rates are the values after warmup.

Item	Value
Training questions / epochs / optimizer updates	14,000 / 1 / 175
Questions per update / rollouts per question	80 / 8
Optimizer	AdamW
Language model / native merger learning rate	1×10^{-6} / 1×10^{-6}
Vision encoder learning rate	2×10^{-7}
Adam betas / epsilon	$(0.9, 0.999)$ / 10^{-8}
Weight decay / gradient norm clip	0.01 / 1.0
Learning-rate schedule	14-update warmup, then constant
Initialization seed	42
Compute / master parameter precision	BF16 / FP32
Parallelism	FSDP; 8 data-parallel ranks
Training hardware	8 NVIDIA A100 80GB PCIe GPUs
Gradient checkpointing	Non-reentrant, enabled for student
Rollout backend	Hugging Face
Training sampling	Temperature 1; top-p 1; no top-k truncation
Maximum response length	1,024 new tokens
Maximum prompt / model length	24,576 / 26,624 tokens
Image pixel limits	65,536 to 16,777,216

Execution details. Training comprises 175 updates and 112,000 generated trajectories; evaluation uses the final checkpoint at update 175. The image processor applies the model's native resize before patchification, with patch size 16 and native spatial merge size 2. Bias, normalization, and parameter tensors with fewer than two dimensions receive zero weight decay.

Each student update microbatch contains up to seven trajectories, and each teacher microbatch up to two. The Hugging Face rollout model is replicated, with a decode-batch cap of 80 and CPU offload between phases. We use Python 3.12.13, PyTorch 2.10.0 (CUDA 12.6), Transformers 5.5.0, FlashAttention 2.8.3, and Ray 2.53.0.

C Training Prompts

We append the task-specific response instruction below to the original question. Braced fields denote runtime substitutions, and the image placeholder denotes a multimodal image input. The chat template fixes the assistant prefix to an empty thinking block followed by an open answer tag:

```
<|im_start|>assistant
<think>

</think>

<answer>
```

Generation stops at the first exact closing tag `</answer>`, with an EOS fallback and the configured length cap. The opening tag belongs to prompt context; the closing tag is sampled, and tokens after stopping do not contribute to the loss.

Task prompt template.

The original question and options are preserved. The task instruction is substituted verbatim from the following list.

```
<image>
{original_question_and_options}

Response requirements: {task_instruction}
```

T1. Multiple choice.

```
Output protocol: Preserve every content and option-label requirement
in the original question. The assistant response is already prefixed
with the exact opening <answer> delimiter. Do not emit another
<answer>. Decide silently, continue with exactly one option label,
then emit exactly one </answer> and stop immediately. Do not output
reasoning, commentary, multiple labels, repetition, or revisions.
```

T2. Mathematics.

```
Output protocol: Preserve every content and internal-format
requirement in the original question. The assistant response is
already prefixed with the exact opening <answer> delimiter. Do not
emit another <answer>. Unless the original question explicitly
requests reasoning or derivation, solve silently and continue with
only the final response. If it explicitly requests reasoning or
derivation, use at most four short steps. Then emit exactly one
</answer> and stop immediately; do not explore alternatives, repeat,
or revise.
```

T3. Numerical answers.

```
Output protocol: Preserve every precision and sign requirement in the
original question. The assistant response is already prefixed with the
exact opening <answer> delimiter. Do not emit another <answer>.
Compute silently, continue with exactly one final scalar number
without units, brackets, or thousands separators, then emit exactly
one </answer> and stop immediately. Do not output reasoning,
commentary, repetition, or revisions.
```

T4. OCR and transcription.

```
Output protocol: Preserve every content and internal-format
requirement in the original question, including requests for
transcription, caption, description, multiple lines, or LaTeX. The
assistant response is already prefixed with the exact opening <answer>
delimiter. Do not emit another <answer>. Continue with the complete
requested response, then emit exactly one </answer> and stop
immediately. Do not add unrelated commentary, repeat, or revise.
```

T5. Regression.

```
Output protocol: Preserve every precision and sign requirement in the
original question. The assistant response is already prefixed with the
exact opening <answer> delimiter. Do not emit another <answer>.
Estimate silently, continue with exactly one final scalar number
without units, brackets, or thousands separators, then emit exactly
one </answer> and stop immediately. Do not output reasoning,
commentary, repetition, or revisions.
```

T6. Open-ended VQA.

```
Output protocol: Preserve every content and internal-format
requirement in the original question. The assistant response is
already prefixed with the exact opening <answer> delimiter. Do not
emit another <answer>. Continue with a concise but complete requested
response, then emit exactly one </answer> and stop immediately. Do not
add unrelated commentary, repeat, or revise.
```

T7. TextVQA.

```
Output protocol: Read the visible text carefully and preserve the
answer wording needed by the question. The assistant response is
already prefixed with the exact opening <answer> delimiter. Do not
emit another <answer>. Continue with the shortest complete answer
supported by the image, then emit exactly one </answer> and stop
immediately. Do not add explanation unless the original question
explicitly requests it; do not repeat or revise.
```

T8. Full-image multiple choice.

```
<image>
Inspect the full image carefully. Use all image regions relevant to
the question, including fine-grained and peripheral details.
Question: {question}
Choices:
A. {option_A}
B. {option_B}
C. {option_C}
D. {option_D}

Response requirements: The assistant response is already prefixed with
the exact opening <answer> delimiter. Decide silently, continue with
exactly one capital option letter (A, B, C, or D), then emit exactly
one </answer> and stop immediately. Do not emit reasoning, commentary,
another <answer>, or any other text.
```

D Evaluation Protocol and Prompt Templates

D.1 Decoding and Scoring

Input and decoding. Evaluation uses the final compressed checkpoint with its exported processor and chat template. The system message is `You are a helpful assistant.` Image inputs precede the dataset-specific text, followed by the same empty-thinking and open-answer assistant prefix used in training. Decoding is greedy, with one response per input, stopping at the first closing-answer tag or the generation cap in Table 10. Questions with multiple images preserve all required image inputs; compression applies to the image spans.

Evaluation implementation. We use Lmms-eval (Zhang et al., 2025a) and VLMEvalKit (Duan et al., 2024) for the chosen benchmarks. V*Bench reports option accuracy. HR-Bench reports a macro average over cycle categories. MMMU reports instance accuracy. MMBench measures circular consistency across option permutations of 1,164 base-question groups. GQA uses normalized exact match, and TextVQA uses EvalAI-style VQA soft accuracy. OCRBench reports a total out of 1,000. POPE reports macro accuracy across adversarial, popular, and random subsets. MME sums per-category scores and paired-question scores, with perception and cognition subtotals.

Table 10: Evaluated splits, row counts, and maximum new tokens. All tasks also use the shared closing-answer-tag stop.

Benchmark	Split / setting	Rows	New tokens
V*Bench	test	191	16
HR-Bench 4K	4K	800	1024
GQA	Testdev balanced	12,578	16
MMMU	validation	900	128
MMBench EN Circular	EN dev (circular)	4,329	1024
POPE	Three COCO subsets	9,000	128
TextVQA	Validation 0.5.1	5,000	1024
OCRBench	test	1,000	128
MME	test	2,374	16

D.2 Evaluation Prompt Templates

The following templates show the question formats and answer instructions; repeated instructions are consolidated for presentation. Option-list fields contain the original ordered choices in `A. value`, `B. value`, and subsequent lines.

E1. V* Bench.

```
{question}
{lettered_options}
Answer with the option's letter from the given choices directly.
```

E2. HR-Bench 4K and 8K.

```
{question.strip()}
{lettered_options_with_trailing_newline}Answer the option letter
directly.
```

E3. MMMU, multiple choice.

```
Question: {question}
Options:
{lettered_options}
Answer with the option letter only.
```

E4. MMMU, open-ended.

```
Question: {question}
Options:
{lettered_options}
Please answer the question directly.
```

This template covers 53 open-ended questions, retaining the Options header even for an empty list; text-only examples remain text-only.

E5. OCRBench.

```
{question.strip()}
```

No additional user instruction or response suffix is added.

E6. MMBench EN Circular.

```
{hint_and_space_if_present}{question} There are several options:
{lettered_options}

Answer with the option's letter from the given choices directly.
```

The template includes the optional hint and valid A–E dataset options.

E7. TextVQA.

```
{question.capitalize()}
Answer the question using a single word or phrase.
```

E8. POPE and GQA.

```
{question.strip()}
Answer the question using a single word or phrase.
```

E9. MME.

```
{question.strip().replace(" Please answer yes or no.", "")}
Answer the question using a single word or phrase.
```

E Efficiency and Training Cost

E.1 Efficiency Measurement

We measure efficiency on 305 TextVQA questions covering 294 images. The subset contains questions whose most frequent human answer has at least six Qwen-tokenizer tokens after lowercasing and whitespace normalization. Each question is measured five times per method using BF16, FlashAttention 2, batch size one, and a common CUDA-graph backend. Decoding replays a shared 128-token ground-truth-conditioned reference with 127 incremental forward passes. Compressed models use the 5% retention rule. HiPrune prunes before the first language-model layer; the EPIC implementation prunes before the third layer and uses a trained adapter merged into the base model.

Request timing runs from prepared CPU tensors to decoded output text, covering transfer, vision encoding, token selection, language-model prefill, reference decoding, and synchronization. Measurements are averaged over repetitions within each question and then across questions. Table 11 defines the resource quantities. We use decimal units: one MB is 10^6 bytes and one TFLOP is 10^{12} operations, with each multiply–accumulate counted as two operations.

Table 11: Resource quantities used in the efficiency evaluation.

Quantity	Definition
LLM prefill FLOPs	Analytical cost of language-model prefill and the final-token output head.
Whole-prefill FLOPs	Analytical cost of vision encoding, selection, and language-model prefill.
Full-attention KV	Key/value tensors in the eight full-attention layers after prefill, including text context.
Total decode state	Full-attention KV plus convolutional and recurrent states in the 24 linear-attention layers.
Replay request time	Elapsed time from prepared CPU tensors to decoded text under the fixed reference replay.

F Explanations of Different training strategies

We further examine different strategies for constructing the full-token teacher during LT-OPD. In all variants, the student generates the on-policy trajectory with compressed visual tokens, while the teacher receives the full visual tokens and provides next-token distributional supervision on the same student-generated prefixes.

Initial-policy teacher. Our default strategy keeps the teacher fixed at the pretrained initialization θ_0 throughout training, i.e., $\pi_T = \pi_{\theta_0}$. This provides a stationary supervision target and prevents the teacher from co-adapting with the compressed student. This strategy is also consistent with prior on-policy self-distillation practice (Zhao et al., 2026).

Trust-region regularization. We also consider an adaptive teacher regularized toward the initial policy. At training step s, the teacher distribution is constructed by interpolating the current full-token policy with the frozen reference policy in log-probability space:

$$\pi_T^{\mathrm{TR}}(a \mid x, s) \propto \exp((1-\lambda)\log \pi_{\theta_0}(a \mid x, s) + \lambda \log \pi_{\theta_s}(a \mid x, s)), \tag{10}$$

where $\lambda \in [0, 1]$ controls how strongly the teacher follows the current policy. This allows the teacher to evolve during training while constraining it from drifting excessively from the pretrained model.

EMA teacher. Alternatively, we maintain a separate teacher whose parameters are updated as an exponential moving average (EMA) of the student:

$$\theta_T^{(s)} = (1-\alpha)\theta_T^{(s-1)} + \alpha\theta_s, \tag{11}$$

with $\theta_T^{(0)} = \theta_0$. The EMA teacher changes more smoothly than the current student and therefore provides a slowly evolving supervision target. Similar EMA regularization has been adopted in recent multimodal on-policy self-distillation (Yuan et al., 2026).

G Ablation on Training Objectives

Table 12 reports ablations results for forward KL, reverse KL, and JSD. Notably, forward KL and reverse KL are two different training objectives described in Yuan et al. (2026). Here we also list the detailed training objectives of them:

$$\text{forward KL} : \mathcal{L}_{\text{FKL}} = \mathbb{E}_{s\sim d_{\pi_{\text{S}}}}\left[D_{\text{KL}}(\pi_{\text{T}}(\cdot \mid s) \,\|\, \pi_{\text{S}}(\cdot \mid s))\right] = \mathbb{E}_{s\sim d_{\pi_{\text{S}}}}\left[\sum_a \pi_{\text{T}}(a \mid s) \log \frac{\pi_{\text{T}}(a|s)}{\pi_{\text{S}}(a|s)}\right],$$

$$\text{Reverse KL} : \mathcal{L}_{\text{RKL}} = \mathbb{E}_{s\sim d_{\pi_{\text{S}}}}\left[D_{\text{KL}}(\pi_{\text{S}}(\cdot \mid s) \,\|\, \pi_{\text{T}}(\cdot \mid s))\right] = \mathbb{E}_{s\sim d_{\pi_{\text{S}}}}\left[\sum_a \pi_{\text{S}}(a \mid s) \log \frac{\pi_{\text{S}}(a|s)}{\pi_{\text{T}}(a|s)}\right].$$

$$\text{JSD} : \mathcal{L}_{\text{JSD}} = \mathbb{E}_{s\sim d_{\pi_{\text{S}}}}\left[\tfrac{1}{2}D_{\text{KL}}(\pi_{\text{T}}(\cdot \mid s) \,\|\, m(\cdot \mid s)) + \tfrac{1}{2}D_{\text{KL}}(\pi_{\text{S}}(\cdot \mid s) \,\|\, m(\cdot \mid s))\right], m(a \mid s) = \frac{\pi_{\text{T}}(a|s)+\pi_{\text{S}}(a|s)}{2}.$$

Table 12: Training-objective results. Full-token and JSD results are retained from the main text. Avg. is the same as Table 2 in main text.

Method	V*Bench	HR-4K	GQA	MMMU	MMB	MME	POPE	TextVQA	OCRB	Avg.
Qwen3.5-4B (*Upper Bound*)	84.3	84.4	60.7	77.6	86.4	2,300.8	89.3	81.2	850	100%
Forward KL ($\text{KL}(p_T \| p_S)$)	70.2	68.8	59.5	**52.2**	**82.2**	1,568.2	**89.1**	62.5	**599**	82.3%
Reverse KL ($\text{KL}(p_S \| p_T)$)	70.2	71.3	**59.6**	51.4	82.0	1,600.4	**89.1**	**63.6**	**599**	**82.8%**
JSD ($\beta = 0.5$)	**74.9**	**73.0**	54.8	50.0	78.9	**2,077.0**	86.3	56.0	535	82.3%

H Ablation on the Visual-Budget Curriculum

The visual-budget schedule keeps the retention at 25% for the first 14 updates, decays it along a cosine curve, and then holds it at 5% for updates 101–175. Table 13 isolates the two constant stages: the *w/o final holding* variant keeps the warmup, the cosine decay, and the 5% endpoint but removes the frozen 5% hold, so the retention reaches 5% only at the last update, while the *w/o initial holding* variant removes the 25% hold so that the decay starts immediately. Both variants otherwise follow the reference training configuration (same initialization seed, data order, and 175-update budget) and are evaluated with the protocol of Table 10 at 5% retention.

Table 13: Effect of removing the two constant stages of the visual-budget schedule. Row *LT-OPD* is the full-schedule model from Table 12. Avg. is the same as Table 2 in main text.

Method	V*Bench	HR-4K	GQA	MMMU	MMB	MME	POPE	TextVQA	OCRB	Avg.
LT-OPD	**74.9**	**73.0**	**54.8**	**50.0**	**78.9**	**2,077.0**	86.3	**56.0**	**535**	**82.3%**
w/o final holding	72.3	69.1	**54.8**	49.8	77.7	2,025.0	**86.4**	55.2	524	80.7%
w/o initial holding	68.1	69.7	53.6	49.6	76.9	1,994.7	84.8	48.3	475	78.0%

I Results on Multi-image Benchmarks

We test on two multi-image benchmarks, MuirBench and MVBench. Note that MuirBench is a multi-image-understanding benchmark, while MVBench is a video-understanding benchmark (16 frames uniform sampling). Table 14 reports the results.

Table 14: Performance on multi-image benchmarks. Two baselines keep 5% visual tokens. Avg. is the same as Table 2 in main text.

Method	MuirBench	MVBench	Avg.
Qwen3.5-4B (*Upper Bound*)	65.7	71.8	100%
CDPruner	39.1	51.6	65.7%
EPIC	44.0	52.2	69.8%
LT-OPD	**52.0**	**55.6**	**78.3%**

J Qualitative Case Studies

We compare HiPrune and EPIC outputs with the final LT-OPD checkpoint on three shared image-question items. We report evidence details in the image, and model predictions.

J.1 Object Material in a Crowded Scene



Evidence detail



Reader-only enlargement of the red stool.

Question. What is the material of the stool?

A. bamboo B. steel C. plastic D. wood

Reference answer. C (plastic)

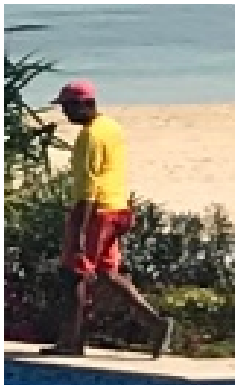
Method	Recorded answer text	Outcome
HiPrune	`A`	Incorrect
EPIC	`D`	Incorrect
LT-OPD	`C`	Correct

Figure 5: V*Bench item 18. A plastic stool appears among boats and produce. HiPrune and EPIC select bamboo and wood, respectively; LT-OPD correctly selects plastic.

J.2 A Small Attribute in a Large Scene



Evidence detail



Question. What is the color of the man's cap?

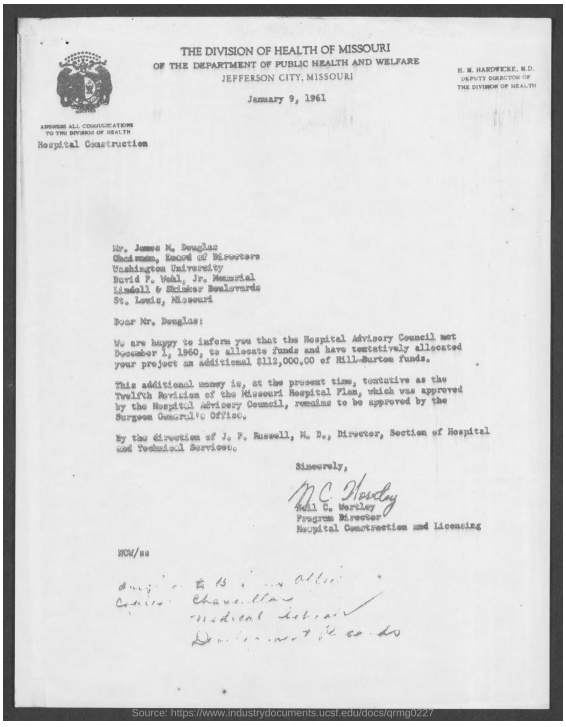
A. white B. red C. gray D. blue

Reference answer. B (red)

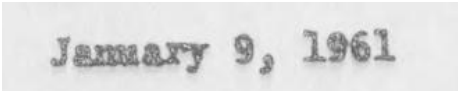
Method	Recorded answer text	Outcome
HiPrune	`(A)`	Incorrect
EPIC	`A`	Incorrect
LT-OPD	`B`	Correct

Figure 6: V*Bench item 29. A red cap occupies a small part of the poolside scene. Both baselines select white; LT-OPD correctly selects red.

J.3 Distinguishing a Document Date from a Mentioned Event Date



Document-date detail



Question. When is the document dated?

Reference answer. January 9, 1961

Method	Recorded answer text	Outcome
HiPrune	`The document is dated **March 13, 1975**.`	Incorrect
EPIC	`The document is dated December 1, 1960.`	Incorrect
LT-OPD	`January 9, 1961`	Correct

Figure 7: OCRBench item 512 (DocVQA). LT-OPD correctly returns the letter's header date, January 9, 1961. EPIC returns the meeting date mentioned in the body, December 1, 1960; HiPrune returns a different date.