LEARNING TO ROUTE PATCHES: MIXTURE OF PATCH EMBEDDINGS VISION TOKENIZATION FOR MLLMS

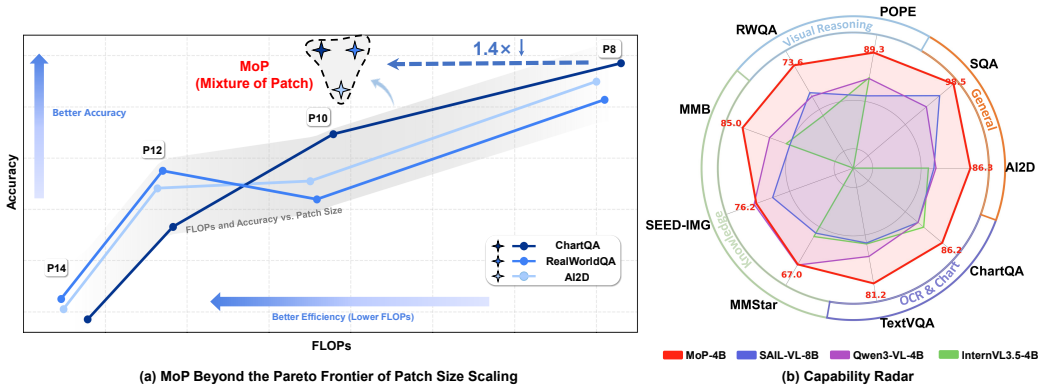
Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models (MLLMs) traditionally rely on a fixed visual tokenization scheme, where all images are discretized using a single patch size, regardless of input complexity or task requirements. This design enforces a uniform trade-off between representation fidelity and computational cost, which we find to be suboptimal. In this work, we investigate the role of visual token granularity under a fixed token budget. Our results show that reducing patch size consistently yields larger performance gains than increasing image resolution at comparable computational cost, indicating that tokenization granularity is a more effective factor in improving multimodal reasoning. However, finer tokenization is not universally optimal, and different benchmarks exhibit distinct performance saturation points, suggesting that the best patch size varies depending on the input and task. Motivated by this, we propose Mixture of Patch Embeddings (MoP), an adaptive visual tokenization framework that jointly trains multiple patch embedding modules and learns a lightweight routing policy to select the most suitable patch size for each input. Extensive experiments across diverse multimodal benchmarks demonstrate that MoP matches the accuracy of the finest fixed patch size while reducing FLOPs by about 1.4× on average, and delivers the best performance on 7 of 9 benchmarks among compared MLLMs. Our results highlight the potential of adaptive tokenization in advancing MLLMs, where the flexibility of patch size selection enhances model performance without necessitating significant efficiency trade-offs.

1 INTRODUCTION

Multimodal large language models (MLLMs) (Bai et al., 2025; Kimi Team et al., 2025; Sun et al., 2025) convert continuous visual signals into discrete tokens for downstream reasoning. In real-world perception, this discretization is constrained by sensing fidelity and computational budgets, requiring rich visual inputs to be encoded into a limited set of representations (Guo et al., 2024; Rao et al.,



(a) MoP Beyond the Pareto Frontier of Patch Size Scaling (b) Capability Radar

Figure 1: **MoP Performance.** (a) MoP Beyond the Pareto Frontier of Patch Size Scaling. (b) MoP delivers consistently strong and balanced performance across diverse vision-language benchmarks.

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.

2021; Bolya et al., 2022). From an information-theoretic perspective, this introduces a trade-off between representation fidelity and encoding cost, which determines how much visual information can be effectively preserved and utilized. In current MLLMs, this encoding is realized through visual tokenization, where images are discretized into patch tokens (Dosovitskiy et al., 2020; Touvron et al., 2021).

Despite its central role, visual tokenization is typically fixed, applying a single patch size to all inputs. Consequently, improvements are often sought by increasing input fidelity, using higher-resolution images (Wang et al., 2025b; Dehghani et al., 2023), while leaving tokenization unchanged. Under limited computational budgets, it remains unclear whether refining the tokenization granularity or increasing input resolution is more effective.

To investigate this, we conduct controlled experiments that isolate the effects of input fidelity and patch size under matched token budgets. We find that reducing patch size generally yields larger performance gains than increasing resolution, indicating that finer-grained tokenization is a more effective mechanism for multimodal reasoning. However, performance is not monotonic: smaller patches are not universally optimal, and saturation points vary across benchmarks and tasks. These observations suggest that the optimal visual tokenization granularity is both input- and task-dependent.

Motivated by this, we propose **Mixture of Patch Embeddings (MoP)**, an adaptive visual tokenization framework that enables instance-wise patch size selection. MoP jointly trains multiple patch embedding modules and a lightweight routing policy, allowing flexible control over representation granularity without increasing the overall token budget.

Evaluations across diverse multimodal benchmarks show that MoP matches the finest fixed patch size at about 1.4× fewer FLOPs and surpasses all coarser fixed patch sizes. Analysis of the learned routing policy further reveals meaningful, task-dependent preferences over tokenization granularity. While Wang et al. (2025a) show a monotonic scaling law forming a Pareto frontier of accuracy versus FLOPs, Figure 1(a) demonstrates that MoP moves beyond the fixed-patch frontier of the same backbone, and Figure 1(b) shows balanced performance across benchmarks, highlighting the effectiveness of adaptive tokenization.

Our contributions are summarized as follows:

- **Empirical insights:** Controlled experiments reveal that patch size scaling is more effective than resolution scaling, but optimal patch size is task- and input-dependent, exhibiting a non-monotonic behavior.
- **Adaptive MoP framework:** Selects patch sizes per input using multiple patch embeddings and a lightweight router.
- **Strong empirical results:** Matches the finest fixed patch size at about 1.4× fewer FLOPs, moving beyond the fixed-patch Pareto frontier with balanced performance across benchmarks.

2 Pilot Experiment

To understand the role of visual tokenization in multimodal reasoning, we conduct controlled experiments under a fixed token budget, examining both input resolution and patch size. Our findings reveal three key phenomena: (i) reducing patch size generally yields larger performance gains than increasing resolution, (ii) there exists an optimal patch size range beyond which gains saturate, and (iii) performance is non-monotonic, with the smallest patch not always yielding the best results. These insights motivate adaptive tokenization to better allocate representation capacity.

2.1 Token Budget Allocation: Resolution vs. Patch Size

Under a standard ViT-style tokenizer, the number of visual tokens is approximately proportional to $\frac{HW}{p^2}$, where H, W denote the input resolution and p denotes the patch size. This enables a controlled comparison between two scaling strategies: **increasing resolution** $(H, W) \to (\lambda H, \lambda W)$ and **decreasing patch size** $p \to p/\lambda$, both of which induce comparable token growth.

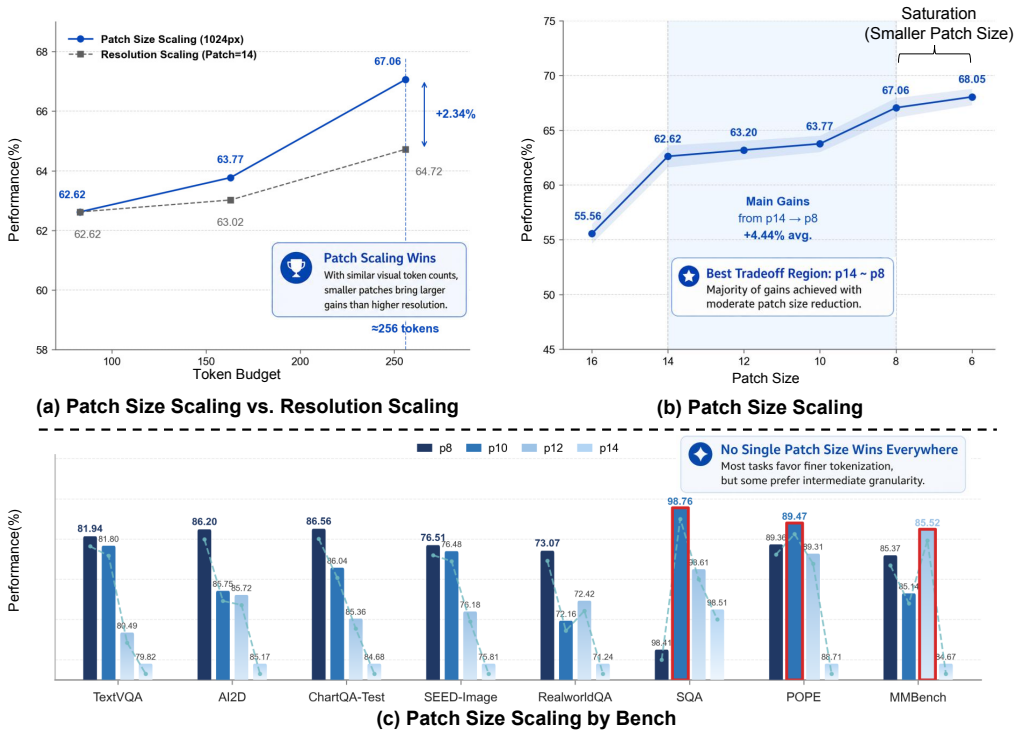


Figure 2: **Patch size scaling Analysis.** (a) At a fixed token budget, reducing patch size consistently yields larger gains than increasing resolution, highlighting token granularity as the key factor. (b) Performance gains are concentrated in the p=14 $\rightarrow$ p=8 regime, with clear saturation at smaller patch sizes. (c) Non-monotonic distribution of optimal patch sizes across different benchmarks.

As shown in Figure 2(a), allocating tokens to finer patch granularity yields larger performance gains than increasing input resolution. This indicates that, under a fixed token budget, the bottleneck lies in the discretization process rather than the raw input fidelity.

A possible explanation is that increasing resolution does not proportionally increase usable information, as additional details must still be compressed into a fixed number of tokens. In contrast, reducing patch size reallocates the token budget toward finer-grained representations, enabling more effective utilization of available information.

2.2 An Efficiency–Performance Sweet Spot

We first examine the global trade-off between performance and computational cost across patch sizes in Figure 2(b). Varying the patch size from $p = 16$ to $p = 6$ yields a consistent efficiency–performance frontier with a favorable operating region of approximately $p \in [8, 14]$: reducing the patch size from $p = 16$ to $p = 14$ brings noticeable gains at minimal additional cost, whereas going from $p = 8$ to $p = 6$ incurs substantially higher computation for only marginal gains.

This pattern indicates diminishing returns as patches become finer. From a global perspective, patch size thus behaves as a controllable trade-off knob, with an optimal efficiency–performance region rather than a single extreme.

2.3 Non-Monotonic and Instance-Dependent Behavior

Despite the existence of a favorable global region, patch size scaling is not monotonic, and no single patch size consistently achieves the best performance across all tasks.

Across benchmarks, the optimal patch size varies significantly, with several tasks peaking at intermediate granularities (e.g., $p = 10$ or $p = 12$) rather than the smallest patch size. As shown in

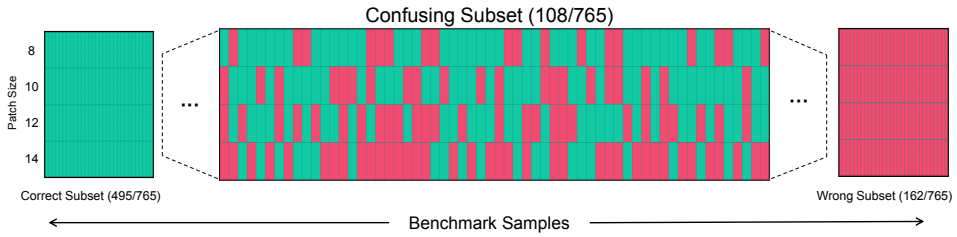


Figure 3: **Instance-dependent patch-size behavior.** Green/red denote correct/incorrect predictions across patch sizes. Most samples are consistently correct or wrong, while the zoomed-in confusing subset shows non-monotonic, sample-specific correctness patterns, motivating adaptive patch-size selection.

Figure 2(c), the distribution of optimal patch sizes differs markedly across benchmarks, and the same non-monotonic behavior also manifests at the input level: Figure 3 visualizes patch-wise correctness across samples, and while most samples are consistently correct or incorrect across all patch sizes, a substantial subset exhibits instance-dependent patterns where the optimal choice varies across inputs.

These results indicate that patch size cannot be treated as a globally optimal hyperparameter. Instead, the optimal choice is inherently input-dependent, and a fixed tokenization strategy leads to suboptimal allocation of representation capacity. This motivates adaptive visual tokenization, where the model dynamically selects the most suitable patch size for each input.

3 METHOD

Our pilot experiments show that finer patch granularity is generally more effective than increasing resolution under a fixed token budget, but the optimal patch size varies across input scenarios. This motivates us to treat visual tokenization as an adaptive routing problem rather than a fixed design choice. As illustrated in Figure 4, we propose **Mixture of Patch Embeddings (MoP)**, an adaptive visual tokenization framework that learns to route each input image to a suitable patch size. Unlike conventional MLLMs that tokenize all images with a fixed patch size, MoP equips a pretrained MLLM with multiple patch embedding branches of different granularities and a lightweight image-conditioned router. The model first adapts these patch-size-specific branches within a shared MLLM backbone, and then learns a routing policy from patch-size-specific performance signals. During inference, only one branch is activated for each image, enabling adaptive tokenization without multi-branch ensembling.

3.1 ADAPTING A PRETRAINED MLLM TO MULTIPLE PATCH SIZES

Let $\mathcal{M}$ denote a pretrained MLLM with a ViT-style visual tokenizer using a single patch embedding layer E_{p_0} with patch size p_0. MoP replaces this fixed tokenizer with a set of patch-size-specific embedding branches $\mathcal{E} = \{E_p \mid p \in \mathcal{P}\}$, where $\mathcal{P} = \{8, 10, 12, 14\}$ in our implementation.

To construct the multi-patch tokenizer, each branch is initialized from the pretrained embedding weights rather than randomly. For a target patch size p, we obtain its initial weight via a FlexiViT-style (Beyer et al., 2023) pseudo-inverse resize:

$$W_p^{(0)} = \text{PI-Resize}_{p_0 \to p}(W_{p_0}), \tag{1}$$

where W_{p_0} is the original embedding weight. This preserves the pretrained representation structure and provides a stable initialization across patch sizes.

We then jointly adapt all branches while sharing the visual encoder, projector, and language model. Given a training set $\mathcal{D} = \{(I_n, Q_n, Y_n)\}_{n=1}^{N}$, where I_n is the input image, Q_n is the textual question or instruction, and Y_n is the target response, the model is optimized using the standard autoregressive language modeling objective. At each training step, only one patch embedding branch is activated, and we cycle through patch sizes in a block-wise manner:

$$p_k = p_{1+(\lfloor k/K \rfloor \bmod M)}, \tag{2}$$

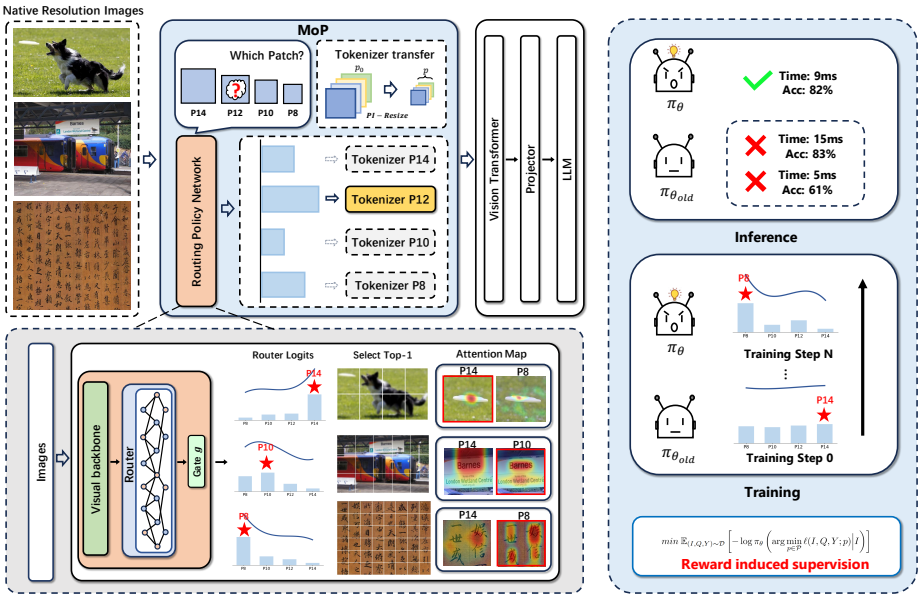


Figure 4: **Overview of the Mixture of Patch Embeddings (MoP) framework.** A lightweight router selects the patch size for each input, which is encoded by the corresponding embedding and fed into a shared multimodal LLM, enabling instance-wise adaptive tokenization.

where $\mathbf{p} = [p_1, \ldots, p_M]$, $M = |\mathcal{P}|$, k is the optimization step, and K is the rotation interval (200 steps in our experiments). This schedule exposes the shared backbone to multiple tokenization granularities while avoiding frequent changes in sequence length, improving training stability and reducing distributed training overhead.

3.2 Patch Routing with Reward-Induced Supervision

After multi-patch adaptation, we train a lightweight router to predict the appropriate patch size for a given image. The router operates before visual tokenization and is conditioned on the input image. Given I, we extract visual features $F = \phi(I)$ using a pretrained backbone $\phi(\cdot)$ (DINOv3-Base in our implementation), followed by global pooling to obtain $\bar{f}$. A two-layer MLP $g_\theta(\cdot)$ produces routing logits $a_\theta(I) = g_\theta(\bar{f}) \in \mathbb{R}^{|\mathcal{P}|}$, which define a categorical policy:

$$\pi_\theta(p|I) = \frac{\exp(a_\theta^p(I))}{\sum_{p' \in \mathcal{P}} \exp(a_\theta^{p'}(I))}. \tag{3}$$

During inference, the router performs top-1 selection. During training, we optionally add small Gaussian noise to the logits to improve robustness and mitigate premature collapse.

To supervise the router, we use the adapted multi-patch MLLM to evaluate the suitability of each candidate patch size. Since the candidate set $\mathcal{P}$ is small, we do not sample patch sizes from the router. Instead, for each training sample, we perform an *exhaustive patch rollout* by evaluating all candidates in $\mathcal{P}$. For a sample (I, Q, Y) and a patch size p, we compute the normalized language modeling loss:

$$\ell(I, Q, Y; p) = -\frac{1}{|Y|} \sum_{m=1}^{|Y|} \log P_{\Theta^*}\left(Y_m \mid Y_{<m}, Q, I; p\right), \tag{4}$$

where Θ^* denotes the adapted multi-patch MLLM. This loss measures how well the model solves the sample when using patch size p. We define a loss-based performance reward as $r(I, Q, Y; p) = -\ell(I, Q, Y; p)$, where higher reward indicates that the corresponding patch size is more suitable for the current sample. Instead of constructing a soft target distribution over patch sizes, we directly identify the optimal patch size as the reward-maximizing candidate:

$p^\star(I, Q, Y) = \arg\max_{p\in\mathcal{P}} r(I, Q, Y; p)$. The router is then trained to predict the optimal patch size by minimizing a cross-entropy objective:

$$\mathcal{L}_{\text{route}} = \mathbb{E}_{(I,Q,Y)\sim\mathcal{D}} \left[-\log \pi_\theta(p^\star(I, Q, Y) \mid I) \right]. \tag{5}$$

Since the teacher signal is obtained by exhaustively evaluating all candidate patch sizes with the adapted MLLM, router training reduces to a supervised classification problem over a small discrete action space. This formulation avoids the variance of sampling-based policy optimization while remaining computationally tractable due to the small size of $\mathcal{P}$. We use the rewards only to rank candidates, since normalizing them into a soft target would mostly encode sample difficulty rather than patch preference.

At inference time, the router deterministically selects the patch size with the highest predicted logit. Formally, given the routing logits $a_\theta(I)$, the selected patch size is:

$$\hat{p}(I) = \arg\max_{p\in\mathcal{P}} a_\theta^p(I), \tag{6}$$

which is equivalent to $\arg\max_p \pi_\theta(p|I)$. The input image is then tokenized using the corresponding patch embedding branch $E_{\hat{p}(I)}$, and only this branch is activated during forward inference. This top-1 routing strategy ensures minimal computational overhead while preserving the adaptivity of visual tokenization.

4 Experiment

4.1 Experimental Setup

Base model. We conduct experiments on Qwen3-VL-4B to validate the effectiveness of the proposed Mixture of Patch Embeddings. Unless otherwise specified, the candidate patch size set is $\mathcal{P} = \{8, 10, 12, 14\}$. All patch embedding branches share the same visual encoder, projector, and language model, while the patch router is trained as an additional lightweight module.

Multi-patch tokenizer adaptation. We adapt the pretrained MLLM to multiple patch sizes with a two-stage training strategy. In the first stage, we use the 4.3M image-text and OCR-captioning samples summarized in Table 5, following the pretraining recipe of LLaVA-UHD-v3 (Sun et al., 2025), to train the visual encoder and projector so that the model can accommodate visual tokens generated by different patch embedding branches. In the second stage, we perform full-parameter instruction tuning on 2.9M samples, which include the 858K SFT data from LLaVA-UHD-v2 (Zhang et al., 2024) together with roughly 2M fine-grained enhancement samples.

In both stages, we adopt the block-wise patch-size rotation strategy described in Sec. 3.1. Specifically, the active patch size is switched every 200 optimization steps. This design exposes the model to different visual tokenization granularities while reducing the training bubbles caused by frequent changes in visual sequence length during distributed training.

Patch router training. After multi-patch tokenizer adaptation, we freeze the entire MLLM and train only the patch router. The adapted MLLM is used as a scoring model to produce patch-size-specific language modeling losses for each training sample. Following Sec. 3.2, the reward-maximizing patch size serves as a hard label for a cross-entropy objective. For router training, we filter the 858K SFT data and keep only single-image samples, so that the routing decision can be clearly associated with one input image (the resulting 20K router set is described in Appendix A.3.2).

Optimization details. We use AdamW for optimization and adopt a cosine learning rate decay schedule. During multi-patch tokenizer adaptation, the learning rate of the visual encoder is set to 1×10^{-5} and that of the language model to 2×10^{-5}. In the subsequent router training stage, where the MLLM is frozen, the learning rate of the router is set to 1×10^{-3}. The same patch-size rotation interval of 200 steps is used throughout both training stages. Other training hyperparameters are kept consistent across different patch-size configurations for fair comparison.

Table 1: **Main results on multimodal benchmarks.** We compare MoP with representative open-source and proprietary MLLMs. All scores are reported as percentages when applicable. Avg. is the mean over the nine benchmarks; entries marked † are source-reported aggregates (or means over the available subset) for models without full coverage and are not directly comparable. Baseline entries are officially reported except those marked ‡, which we evaluated under the protocol of Appendix A.2; all MoP entries are self-evaluated.

Model	Scale	Avg.	MMBench	SEED-Img	POPE	RealWorldQA	SQA	AI2D	MMStar	TextVQA	ChartQA
Closed-source MLLMs											
GPT-4V (Achiam et al., 2023)	–	68.8†	79.8	–	–	–	–	78.6	56.0	–	–
Claude-3.7-Sonnet (Anthropic, 2025)	–	74.4†	79.7	–	–	–	–	82.5	65.1	–	–
Gemini-1.5-Pro (Gemini Team, 2024)	–	74.8†	73.9	–	–	–	–	83.3	67.1	–	–
Open-source MLLMs											
Kimi-VL (Kimi Team et al., 2025)	3B	–	83.1	–	–	68.1	–	84.9	61.3	–	–
MiniCPM-V-4 (Yu et al., 2025)	4B	76.2†	79.7	–	–	–	–	82.9	61.9	–	–
InternVL3.5 (Wang et al., 2025c)	4B	77.8†	80.3	–	88.9	66.3	–	82.6	65.0	77.9	86.0
Ovis2 (Lu et al., 2025)	4B	76.3†	81.4	–	–	–	–	85.7	61.9	–	–
LLaVA-OneVision (Li et al., 2024a)	7B	77.4†	80.8	75.4	–	66.3	96.0	81.4	61.7	–	80.0
SAIL-VL (Yin et al., 2025)	8B	80.1	79.5	75.5	85.2	71.9	98.2	83.7	64.2	77.7	84.6
Baseline and our model (Qwen3-VL-4B backbone)											
Qwen3-VL-Instruct (Bai et al., 2025)	4B	81.3	83.9	**77.4**‡	88.7‡	71.0‡	91.8‡	84.1	**69.8**	80.4‡	84.6
MoP (ours)	**4B**	**82.6**	**85.0**	76.2	**89.3**	**73.6**	**98.5**	**86.3**	67.0	**81.2**	**86.2**

Evaluation benchmarks. We evaluate MoP on a diverse set of multimodal benchmarks covering general visual understanding, visual reasoning, OCR, document and chart understanding, and hallucination evaluation. Specifically, we report results on MMBench (Liu et al., 2024), SEED-Img (Li et al., 2024b), POPE (Li et al., 2023), RealWorldQA (XAI, 2024), ScienceQA (SQA) (Lu et al., 2022), AI2D (Kembhavi et al., 2016), MMStar (Chen et al., 2024), TextVQA (Singh et al., 2019), and ChartQA (Masry et al., 2022). For each benchmark, we follow the official evaluation protocol and report the corresponding standard metric.

4.2 Main Results

Table 1 compares MoP with representative open-source and proprietary MLLMs. With only a 4B backbone, MoP achieves an average score of 82.6, showing strong overall performance across general multimodal benchmarks. Compared with the Qwen3-VL-Instruct backbone, MoP improves seven of the nine tasks; the exceptions are SEED-Img, where MoP trails by 1.2 points, and MMStar, whose baseline entry is officially reported whereas ours is self-evaluated (the same baseline scores 60.7 under our protocol, Table 2). For a controlled comparison under a single protocol, Table 2 re-evaluates Qwen3-VL-4B with our offline pipeline against MoP trained on the same data, and every ablation in the appendix follows this self-evaluated protocol. MoP is also competitive with larger models: it outperforms LLaVA-OneVision-7B on most reported benchmarks, matches or exceeds several recent 7B–8B MLLMs on visual reasoning and general understanding, and remains competitive with proprietary models on POPE, SQA, AI2D, MMStar, and TextVQA despite a substantially smaller open-source backbone. These results support our motivation that fixed patch tokenization is suboptimal.

4.3 Ablation Studies

Module ablation. Figure 5 studies the effects of multi-patch training and adaptive routing. Starting from Qwen3-VL-4B, multi-patch training improves RealWorldQA, AI2D, MMStar, and ChartQA across different fixed patch sizes, suggesting better visual representation under multiple granularities. Meanwhile, the best patch size varies across benchmarks, $p{=}14$ performs best on MMStar while $p{=}8$ is strongest on RealWorldQA, AI2D and ChartQA, consistent with the pilot finding that no single granularity is universally optimal. By introducing the MoP router, the model adaptively selects patch sizes per input and achieves the best results on RealWorldQA, MMStar and AI2D while remaining competitive on ChartQA, showing that MoP benefits from both multi-patch training and instance-adaptive routing.

Data ablation. To verify that MoP's gains do not simply come from additional training data, we conduct a data-controlled ablation in Table 2. We continue training the original Qwen3-VL-4B

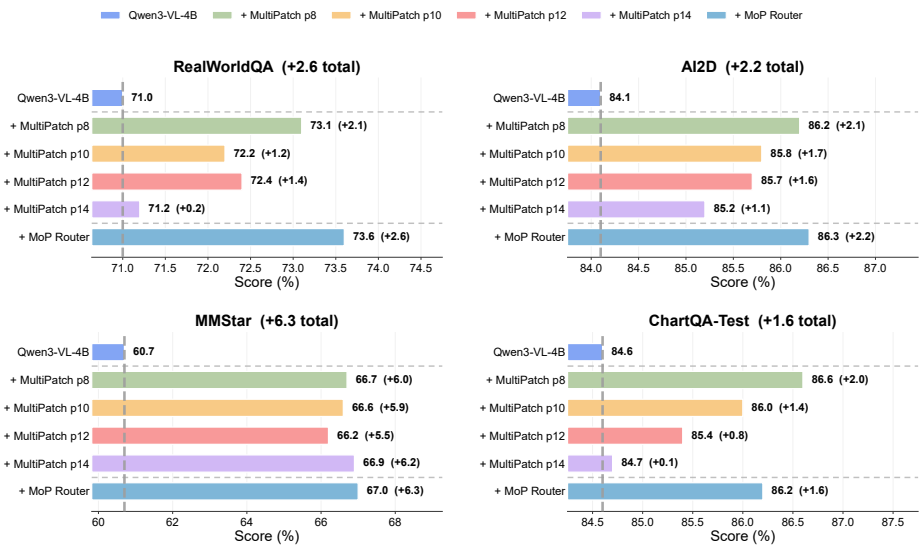


Figure 5: **Module ablation of multi-patch adaptation and the MoP router.** We compare Qwen3-VL-4B with fixed multi-patch models and the adaptive router on four benchmarks. Parentheses denote absolute gains over the baseline.

Table 2: **Data ablation.** The data-only baseline uses the same training data as MoP but keeps the original Qwen3-VL architecture unchanged. Avg. is computed over the six benchmarks shown.

Model	Training Data	Structure	Routing	Avg.	MMBench	POPE	RealWorldQA	MMStar	TextVQA	ChartQA
Qwen3-VL-4B	–	original	–	77.6	81.9	88.7	71.0	60.7	80.4	82.8
Qwen3-VL-4B	~5M	original	–	77.8	81.9	87.5	72.8	62.0	78.5	83.9
MoP (ours)	~5M	multi-patch	adaptive	**78.9**	**83.0**	**89.3**	**72.9**	**62.9**	80.0	**85.4**

architecture on the same data used by MoP while keeping its visual tokenization unchanged, and re-evaluate all models under the same offline evaluation framework. The data-only baseline brings only marginal improvement over the re-evaluated Qwen3-VL-4B baseline, increasing the average score from 77.6 to 77.8. In contrast, MoP achieves a much higher average score of 78.9 and consistently outperforms the data-only baseline across multiple tasks, demonstrating that its gains cannot be attributed to additional data alone, but rely on the multi-patch architecture and adaptive routing.

Backbone generalization. Applying the complete MoP framework to LLaVA-OneVision-7B raises its average score from 76.8 to 79.5 under the same offline protocol, with gains on eight of the nine benchmarks. Reusing the router trained on Qwen3-VL instead reaches 79.2 and trails on five of the nine benchmarks, indicating that patch-size preferences are partly model-specific and that the router is best trained on the target backbone (Appendix A.5.1).

4.4 Routing Behavior Analysis

Achievement Rate on Confusing Samples. To assess instance-level effectiveness, we focus on *confusing samples*, where at least one fixed patch size answers correctly while another fails. The achievement rate measures whether the router-selected patch falls within the set of correct choices. As shown in Table 3, MoP consistently outperforms a random patch-selection baseline and exceeds the best fixed patch on AI2D, RealWorldQA and MMStar. This indicates that the router learns non-trivial instance-specific routing rather than following a global resolution preference. The learned router also outperforms the heuristic AdaPatch strategy (Liu et al., 2026) when both select patch sizes for the same multi-patch backbone (82.6 vs. 79.9 on average; Appendix A.5.2).

Router Selection Diversity. Figure 6 visualizes the router's patch-size choices across benchmarks. The routing behavior is clearly dataset-dependent: AI2D and MMStar contain more samples routed

Table 3: **Achievement rate on confusing samples.** We evaluate whether the selected patch size produces a correct answer on samples where different fixed patch sizes lead to different correctness outcomes. Random denotes the expected accuracy of uniformly sampling one patch size from $\{8, 10, 12, 14\}$ for each sample. All numbers are percentages.

Benchmark	# Samples	Random	p=8	p=10	p=12	p=14	Router	Δ vs. Best Fixed
AI2D	253	50.4	56.5	51.0	49.8	42.7	**58.1**	+1.6
RealWorldQA	108	52.3	62.0	54.6	51.9	40.7	**63.0**	+0.9
MMStar	225	49.8	51.1	50.2	45.8	52.0	**52.9**	+0.9

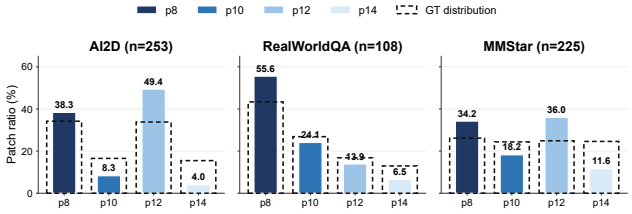


Figure 6: **Patch-size selection distribution on confusing samples.** The router shows dataset-dependent preferences, favoring p=12 on AI2D and MMStar and p=8 on RealWorldQA. GT denotes the oracle distribution of correct patch sizes.

to p=12, whereas RealWorldQA favors p=8 more frequently. Rather than implying that p=12 is globally superior on AI2D or MMStar, the router appears to reserve smaller patch sizes such as p=8 for samples that critically rely on fine-grained visual details, while assigning larger patch sizes to samples that remain solvable across multiple resolutions. This suggests a difficulty-aware allocation strategy rather than a match to benchmark-level statistics. Nevertheless, as Table 3 shows, accurate instance-level routing remains essential, especially for benchmarks with strong fixed-patch biases.

5 Related Work

MLLMs inherit the ViT recipe of patchifying images with a single fixed patch size (Dosovitskiy et al., 2020); most efforts to improve visual fidelity act on the input side by supporting arbitrary resolutions and aspect ratios (Dehghani et al., 2023; Fan et al., 2024; Wang et al., 2024; Guo et al., 2024). FlexiViT (Beyer et al., 2023) makes one backbone operate at multiple patch sizes, and patchification scaling laws (Wang et al., 2025a) report monotonic gains from smaller patches, but in both the patch size stays a hand-chosen global hyperparameter. Adaptive token budgets have instead been obtained by pruning or merging tokens after encoding (Rao et al., 2021; Bolya et al., 2022; Sun et al., 2025), by nested representations that expose several token counts (Cai et al., 2025; Hu et al., 2024), or by heuristic adaptive patching (Liu et al., 2026; Choudhury et al., 2025). MoP instead sets the discretization granularity before encoding, which is complementary to token compression, and learns the choice from reward-ranked rollouts rather than hand-designed rules (Appendix A.5.2). Its routing is inspired by mixture-of-experts (Shazeer et al., 2017; Riquelme et al., 2021; Zhou et al., 2022) and mixtures of vision encoders (Zong et al., 2024), but routes among tokenizer granularities of one shared encoder and activates a single branch.

6 Conclusion

We identify patch size as a crucial and underexplored scaling dimension for vision-language models. Our study demonstrates that patch-size scaling can outperform resolution scaling with comparable token budgets, though its effects are not strictly monotonic across benchmarks, indicating that different inputs may benefit from varying granularities. Based on this insight, we propose Mixture of Patch Embeddings, an adaptive framework with a lightweight router that selects the optimal patch size for each input. Experiments show that our method matches the accuracy of the finest fixed patch size at about 1.4× fewer FLOPs, offering improved accuracy-efficiency trade-offs over fixed patch-size baselines.

AI USE STATEMENT

In this work, we used generative AI tools for assisting with the writing and debugging of data processing scripts, including the scripts that aggregate raw evaluation outputs into the reported statistics. We have not used generative AI tools for generating research ideas or hypotheses, designing experiments, implementing the MoP method or its training and evaluation code, interpreting experimental results, or translating any part of the manuscript, and formal proofs are not applicable to this work. Additionally, we used generative AI tools for polishing the writing and checking grammar of author-written text without altering its content, for assisting with the plotting scripts that render the data figures (all schematic figures were drawn manually by the authors), and for minor LaTeX formatting assistance. We have reviewed all AI-assisted work: all AI-assisted scripts were verified and tested for correctness by the authors before use, and all AI-polished text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe the MoP architecture and the two-stage training procedure in Section 3 and the experimental setup in Section 4.1. Implementation details of the pilot experiments, module and data ablations, routing-behavior analysis, data curation, training recipe, and the offline VLMEvalKit evaluation protocol (including decoding hyperparameters) are provided in the appendix. Anonymized source code for multi-patch adaptation, router training, and evaluation is provided as supplementary material.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic. Claude 3.7 sonnet, 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

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.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14496–14506, 2023.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. In *International Conference on Learning Representations*, 2025.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models?, 2024. URL https://arxiv.org/abs/2403.20330.

Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, László A Jeni, and Kris M Kitani. Accelerating vision transformers with adaptive patch sizes. *arXiv preprint arXiv:2510.18091*, 2025.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution. *Advances in Neural Information Processing Systems*, 36:2252–2274, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Qihang Fan, Quanzeng You, Xiaotian Han, Yongfei Liu, Yunzhe Tao, Huaibo Huang, Ran He, and Hongxia Yang. Vitar: Vision transformer with any resolution. *arXiv preprint arXiv:2403.18361*, 2024.

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *arXiv preprint arXiv:2503.21776*, 2025.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

Jarvis Guo, Tuney Zheng, Yuelin Bai, Bo Li, Yubo Wang, King Zhu, Yizhi Li, Graham Neubig, Wenhu Chen, and Xiang Yue. Mammoth-vl: Eliciting multimodal reasoning with instruction tuning at scale, 2025. URL https://arxiv.org/abs/2412.05237.

Zonghao Guo, Ruyi Xu, Yuan Yao, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. In *European Conference on Computer Vision*, pp. 390–406. Springer, 2024.

Wenbo Hu, Zi-Yi Dou, Liunian Harold Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka query transformer for large vision-language models. *Advances in Neural Information Processing Systems*, 2024.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

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.

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

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13299–13308, 2024b.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.

Wenzhuo Liu, Weijie Yin, Fei Zhu, Shijie Ma, Haiyang Guo, Xiao-Hui Li, Cheng-Lin Liu, et al. One patch doesn't fit all: Adaptive patching for native-resolution multimodal large language models. In *The Fourteenth International Conference on Learning Representations*, 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 *European conference on computer vision*, pp. 216–233. Springer, 2024.

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

Shiyin Lu, Yang Li, Yu Xia, Yuwei Hu, Shanshan Zhao, Yanqing Ma, Zhichao Wei, Yinglun Li, Lunhao Duan, Jianshan Zhao, et al. Ovis2. 5 technical report. *arXiv preprint arXiv:2508.11737*, 2025.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning, 2022. URL https://arxiv.org/abs/2203.10244.

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

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

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.

Shichu Sun, Yichen Zhang, Haolin Song, Zonghao Guo, Chi Chen, Yidan Zhang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Llava-uhd v3: Progressive visual compression for efficient native-resolution encoding in mllms. *arXiv preprint arXiv:2511.21150*, 2025.

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

Feng Wang, Yaodong Yu, Guoyizhe Wei, Wei Shao, Yuyin Zhou, Alan Yuille, and Cihang Xie. Scaling laws in patchification: An image is worth 50,176 tokens and more, 2025a. URL https://arxiv.org/abs/2502.03738.

Fengxiang Wang, Mingshuo Chen, Yueying Li, Di Wang, Haotian Wang, Zonghao Guo, Zefan Wang, Boqi Shan, Long Lan, Yulin Wang, et al. Geollava-8k: Scaling remote-sensing multimodal large language models to 8k resolution. *arXiv preprint arXiv:2505.21375*, 2025b.

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

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*, 2025c.

XAI. RealWorldQA, September 2024. URL https://huggingface.co/datasets/xai-org/RealworldQA. Accessed: 2024-09-24.

Weijie Yin, Yongjie Ye, Fangxun Shu, Yue Liao, Zijian Kang, Hongyuan Dong, Haiyang Yu, Dingkang Yang, Jiacong Wang, Han Wang, et al. Sail-vl2 technical report. *arXiv preprint arXiv:2509.14033*, 2025.

Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Yuanqian Zhao, et al. Minicpm-v 4.5: Cooking efficient mllms via architecture, data, and training recipe. *arXiv preprint arXiv:2509.18154*, 2025.

Xiaoying Zhang, Da Peng, Yipeng Zhang, Zonghao Guo, Chengyue Wu, Jen-Tse Huang, Chi Chen, Wei Ke, Helen Meng, and Maosong Sun. Will pre-training ever end? a first step toward next-generation foundation mllms via self-improving systematic cognition. *arXiv preprint arXiv:2503.12303*, 2025.

Yipeng Zhang, Yifan Liu, Zonghao Guo, Yidan Zhang, Xuesong Yang, Xiaoying Zhang, Chi Chen, Jun Song, Bo Zheng, Yuan Yao, et al. Llava-uhd v2: an mllm integrating high-resolution semantic pyramid via hierarchical window transformer. *arXiv preprint arXiv:2412.13871*, 2024.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Zhuofan Zong, Bingqi Ma, Dazhong Shen, Guanglu Song, Hao Shao, Dongzhi Jiang, Hongsheng Li, and Yu Liu. MoVA: Adapting mixture of vision experts to multimodal context. *Advances in Neural Information Processing Systems*, 2024.

A TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A.1 IMPLEMENTATION DETAILS OF EXPERIMENTS

A.1.1 PILOT EXPERIMENT SETTING

Both Figure 2(a) and Figure 2(b) in the main paper are conducted based on LLaVA-UHD-v3 (Sun et al., 2025). All model variants follow the same training recipe, including Stage-1 pretraining on 4.558M samples and Stage-2 supervised fine-tuning on 858K samples. Evaluation is performed across eight multimodal benchmarks: MMBench, SEED-Img, POPE, RealWorldQA, SQA, AI2D, TextVQA, and ChartQA, and the curves in Figure 2(a) and (b) report the average over these eight benchmarks.

A.1.2 MODULE ABLATION SETTING

For the module ablation in Figure 5 of the main paper, we train a multi-patch backbone from Qwen3-VL-4B using 4.558M pretraining samples and 3M OCR-oriented training samples. This ablation uses a different data recipe from the main model in Table 5, and is therefore only compared against settings trained under the same recipe. The trained backbone is evaluated by fixing the patch size to each candidate branch. We further collect per-patch forward losses on 20K samples from the 3M training set and use them to train the lightweight router. The final MoP model is obtained by attaching the trained router to the multi-patch backbone. All settings are evaluated under the same offline VLMEvalKit protocol.

A.1.3 DATA ABLATION SETTING

For the data ablation, we adopt a lighter training recipe with 4.558M pretraining samples and 858K SFT samples. We first continue training the original Qwen3-VL-4B baseline on the same data while keeping its architecture and visual tokenization scheme unchanged. We then train MoP under the same data setting, where both the multi-patch backbone and the lightweight router are introduced. By comparing these two variants, we isolate the effect of the proposed architectural design from that of additional training data. All training hyperparameters are kept consistent with those described in the main paper.

A.1.4 ROUTING BEHAVIOR ANALYSIS SETTING

We analyze the routing behavior based on the trained multi-patch backbone. For each benchmark sample, we evaluate the model by fixing the patch size to each candidate branch $p \in \mathcal{P} = \{8, 10, 12, 14\}$, and record whether the prediction is correct under each patch size. Since the router only selects one branch from the same multi-patch backbone, its output is restricted to the union of predictions produced by these fixed patch sizes. Therefore, samples that are correctly answered by all patch sizes or incorrectly answered by all patch sizes do not provide meaningful evidence for routing decisions. We remove these samples and focus on the remaining *confusing samples*, where different patch sizes lead to different correctness outcomes. This setting directly evaluates whether the router can select an effective patch size when the choice matters.

Formally, given a benchmark set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ and the prediction $\hat{y}_i^p$ under patch size p, we define the correctness indicator as

$$c_i^p = \mathbb{I}\left[\mathrm{Eval}(\hat{y}_i^p, y_i) = 1\right].$$

The confusing subset is then defined as

$$\mathcal{C} = \left\{ i \mid 0 < \sum_{p \in \mathcal{P}} c_i^p < |\mathcal{P}| \right\}.$$

For a router $r_\theta(x_i) \in \mathcal{P}$, we measure its achievement rate on confusing samples as

$$\mathrm{Achieve}(r_\theta) = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} c_i^{r_\theta(x_i)}.$$

We also report the random-selection baseline,

$$\text{Achieve}_{\text{rand}} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} c_i^p,$$

which corresponds to uniformly sampling one patch size from $\mathcal{P}$ for each sample. A higher achievement rate indicates that the router more often selects a patch size that yields a correct answer on samples where different patch sizes disagree.

Fixed-patch accuracy versus routing frequency. Table 3 and Figure 6 measure different quantities and should not be read as contradicting each other. Table 3 reports the accuracy obtained when one fixed patch size is applied to *all* confusing samples, whereas Figure 6 shows the oracle distribution of the best patch size over *individual* samples. A patch size with lower global accuracy can still be the best choice for a considerable subset of samples; on AI2D, for instance, p=12 achieves a lower fixed-patch accuracy than p=8 yet is preferred by many samples, indicating that it provides useful granularity for specific cases. This sample-level diversity of patch preference is exactly what motivates adaptive selection, and the similarity between the router selections and the oracle distribution shows that MoP learns to route samples to their suitable patch sizes.

A.2 Implementation detail of Evaluation

For the main comparison, we report officially reported results for existing baseline models whenever available. For the Qwen3-VL-4B baseline, MMBench, AI2D, MMStar, and ChartQA are taken from the official report, while SEED-Img, POPE, RealWorldQA, SQA, and TextVQA are evaluated by ourselves. Consequently, the baseline row of Table 1 mixes officially reported and self-evaluated entries, and is intended to position the absolute capability of MoP rather than to serve as a controlled comparison. For all ablation studies in this appendix, by contrast, every compared setting is re-evaluated by ourselves under the same offline VLMEvalKit protocol, so the numbers are internally consistent but may differ from the officially reported values in Table 1. The decoding hyperparameters used in our offline evaluation are summarized in Table 4; decoding is greedy and therefore deterministic, so all self-evaluated numbers come from a single run.

Table 4: Decoding hyperparameters used in our offline VLMEvalKit evaluation.

Hyperparameter	Value
Decoding	Greedy (`do_sample = False`)
Max new tokens	2048
Repetition penalty	1.0

A.3 Data Curation

A.3.1 Data Curation for Pretrained MLLM to Multiple Patch Sizes

For data curation, we adopt a two-stage training recipe summarized in Table 5. In Stage 1, we use 4.3M image-text and OCR-captioning samples to adapt the visual encoder and projector to multi-patch tokenization. The data mainly consists of large-scale image-text pairs from LLaVA-Pretrain and LLaVA-Recap, together with OCR-oriented captioning data from SynthDoG and UReader. In Stage 2, we use 2.9M instruction-following and reasoning samples for supervised fine-tuning. This stage combines coarse-detailed data from MammothVL, LLaVA-SFT, reasoning data, detailed captioning data, and OCR/chart/document-oriented samples to improve the model's general multimodal instruction-following ability.

A.3.2 Data Curation for router

The 20K training set for the router is constructed from the 858K SFT dataset used in model pretraining, where each sample is augmented with patch-wise performance signals (e.g., per-patch losses or derived optimal patch sizes). We first perform strict data cleaning to ensure supervision quality: we retain only samples with exactly one image, a single human query, and a non-empty textual input,

Table 5: Composition of training data across different stages.

Stage	Dataset	Type	#Samples	Total
Stage 1	LLaVA-Pretrain	Image-Text Pairs	558K	4.3M
	LLaVA-Recap-118K	Image-Text Pairs	118K	
	LLaVA-Recap-558K	Image-Text Pairs	558K	
	LLaVA-Recap-CC3M	Image-Text Pairs	2.8M	
	SynthDoG-EN/ZH	OCR Caption	200K	
	UReader-TR-Processed	OCR Caption	101K	
Stage 2	Coarse-Detailed Data from MammothVL (Guo et al., 2025)	Instruction-Following	1.6M	2.9M
	LLaVA-SFT-858K	Instruction-Following	858K	
	Video-R1-CoT (Feng et al., 2025)	Reasoning	80K	
	SiCOG (Zhang et al., 2025)	Detailed Image Caption, Reasoning	70K	
	PixMo	OCR, Chart, Doc	310K	

Table 6: Training configuration for MoP.

Config	Stage 1	Stage 2	Stage 3
Data type	Image-text pairs, OCR caption	OCR, document, chart, detailed captions, interleaved data, pure text corpus	patch-wise aware SFT data
Data volume & batch size	4.3M, 256	2.9M, 128	20K, 512
Trainable parts & LR	ViT: 1e-5 multi-patch tokenizer: 1e-5	ViT/multi-patch tokenizer: 1e-5 LLM: 2e-5	Router: 1e-3

and discard multi-image, multi-turn, or pure-text instances; we further filter out samples whose question length exceeds 40 tokens to control linguistic complexity. For samples associated with the same image, we apply image-level deduplication by keeping only the instance with the shortest query. Each sample is then assigned to a semantic group based on its image source, covering general reasoning (e.g., LLaVA-style, GQA, VG), natural image understanding (e.g., COCO, ADE20K), OCR and TextVQA (e.g., TextVQA, TextOCR, COCO-Text), chart and document understanding (e.g., ChartQA, DocVQA, InfographicVQA), and science/mathematical reasoning (e.g., ScienceQA, AI2D, CoSyn). The final dataset is constructed via balanced sampling across these groups with predefined proportions (30%, 20%, 15%, 15%, 15%, and 5% for supplementary data), while additionally enforcing per-source upper bounds to avoid dominance of large-scale corpora. During sampling, we preserve the global distribution of optimal patch sizes by allocating samples proportionally across patch categories, and prioritize shorter queries within each source to reduce noise. If certain groups cannot meet their target quotas due to filtering or source constraints, the remaining budget is filled with valid samples from other groups without violating source or patch-size balance. This procedure yields a diverse and well-balanced training set that provides stable and informative supervision for learning the routing policy.

A.4 Training Recipe

Table 6 summarizes the training recipe of MoP. In Stage 1, we train the visual encoder and multi-patch tokenizer using 4.3M image-text and OCR-captioning samples to adapt the model to mixed patch granularities. In Stage 2, we perform full-parameter supervised fine-tuning on the curated 2.9M instruction-following, reasoning, detailed captioning, and OCR-related data. In Stage 3, we freeze the trained multi-patch backbone and train the lightweight router with 20K samples, where the supervision is derived from the collected per-patch forward losses.

A.5 Additional Experiments

A.5.1 Generalization to Other MLLM Backbones

To verify that the improvements of MoP are not tied to a specific backbone, we apply the complete MoP framework to LLaVA-OneVision-7B (Li et al., 2024a), and use this setting to examine whether the router itself is transferable across backbones. Table 7 reports three settings in increasing order

Table 7: **Generalization of MoP to LLaVA-OneVision-7B.** All rows are re-evaluated by ourselves under the same offline protocol and are therefore mutually comparable. "+ Qwen3-VL Router" is an intermediate setting that applies the router trained with Qwen3-VL supervision to the LLaVA-OneVision multi-patch model without re-training; "+ MoP" is the full method with the router trained on LLaVA-OneVision's own patch-size preferences.

Model	Avg.	MMBench	SEED-Img	POPE	RealWorldQA	SQA	AI2D	MMStar	TextVQA	ChartQA
LLaVA-OV-7B	76.8	80.2	**75.7**	89.1	65.3	91.1	79.6	57.6	72.2	80.0
LLaVA-OV-7B + Qwen3-VL Router	79.2	84.6	75.2	89.2	65.1	**97.6**	80.5	**59.5**	80.4	**81.5**
LLaVA-OV-7B + MoP	**79.5**	**85.1**	75.1	**89.7**	**65.7**	**97.6**	**81.2**	59.3	**80.6**	81.4

Table 8: **Comparison with AdaPatch under the same multi-patch adapted Qwen3-VL backbone.** Both settings are re-evaluated by ourselves under the same offline protocol. Δ denotes MoP minus AdaPatch.

Method	Avg.	MMBench	SEED-Img	POPE	RealWorldQA	SQA	AI2D	MMStar	TextVQA	ChartQA
AdaPatch	79.9	83.4	**77.4**	89.0	71.1	93.0	80.8	61.6	80.3	82.8
MoP (ours)	**82.6**	**85.0**	76.2	**89.3**	**73.6**	**98.5**	**86.3**	**67.0**	**81.2**	**86.2**
Δ	+2.7	+1.6	-1.2	+0.3	+2.5	+5.5	+5.6	+5.4	+0.9	+3.4

of adaptation: the original model, the LLaVA-OneVision multi-patch model equipped with the router trained on Qwen3-VL, and the full MoP whose router is trained on LLaVA-OneVision's own patch-size loss distribution.

The intermediate setting isolates the router. Directly reusing the Qwen3-VL router, without any re-training, still improves the average score from 76.8 to 79.2. Part of this gain comes from the multi-patch backbone itself, and part from the fact that the transferred router captures general visual complexity, so it remains a reasonable zero-shot choice. However, it falls short of the backbone-specific router on average (79.2 vs. 79.5) and on five of the nine benchmarks, including MMBench, POPE, RealWorldQA, and AI2D, while being marginally ahead on SEED-Img, MMStar, and ChartQA. The margin is small, which is expected since both routers operate on the same multi-patch backbone and share the same notion of visual complexity. Still, the two routers see identical images and differ only in which model's losses supervised them, so the consistent direction of the gap indicates that patch-size preferences are partly model-specific: the granularity that minimizes the loss of one MLLM is not always the best for another.

The full MoP, with the router trained on LLaVA-OneVision's own preferences, gives the final result: 79.5 on average with gains on eight of the nine benchmarks over the original model. This confirms that MoP generalizes across model scales and architectures, and that the router should be trained on the target backbone rather than transferred, which is cheap given that router training only requires 20K samples and keeps the MLLM frozen.

A.5.2 Comparison with Heuristic Adaptive Patching

Existing adaptive patching methods adopt different base models, and some routing strategies rely on heuristic rules rather than learned routing networks. For a fair comparison, we apply the AdaPatch routing strategy (Liu et al., 2026) to the same multi-patch adapted Qwen3-VL backbone used by MoP, so that the two methods differ only in how the patch size is selected. As shown in Table 8, even with only 20K samples for router training, the loss-supervised learned router achieves stronger overall performance than heuristic adaptive patch selection (82.6 vs. 79.9 on average), with particularly large margins on SQA, AI2D, and MMStar.

A.5.3 Ablation on the Router's Visual Encoder

The router is designed to estimate the visual granularity required by an image rather than its semantic content, so a self-supervised vision foundation model is a natural choice for its feature extractor: language-supervised models such as CLIP (Radford et al., 2021) are optimized for semantic alignment and may not fully capture the fine-grained visual complexity relevant to patch-size selection. We compare DINOv3-Base (Siméoni et al., 2025), CLIP-Base, and DINOv3-Small as the router backbone,

Table 9: **Ablation on the router's visual encoder.** All settings are re-evaluated by ourselves under the same offline protocol, so the absolute values differ slightly from Table 1.

Router encoder	Overall	MMBench	SEED-Img	POPE	RealWorldQA	SQA	AI2D	MMStar	TextVQA	ChartQA
DINOv3-Base	**83.4**	**89.3**	**76.3**	90.7	**73.9**	98.5	**86.3**	**67.3**	80.8	87.8
CLIP-Base	83.2	89.2	**76.3**	**90.9**	73.2	**98.6**	85.4	66.9	80.6	87.6
DINOv3-Small	83.2	89.2	**76.3**	90.8	72.4	98.5	**86.3**	66.7	**80.9**	**87.9**

Table 10: **Computational cost and routing distribution of different router encoders.** Avg. TFLOPs is averaged over all benchmarks; the last four columns give the fraction of samples routed to each patch size.

Router encoder	Overall	Avg. TFLOPs	p=8	p=10	p=12	p=14
DINOv3-Base	**83.4**	**9.80**	37.4%	20.5%	31.0%	11.1%
CLIP-Base	83.2	9.83	37.6%	26.9%	19.0%	16.4%
DINOv3-Small	83.2	10.33	45.3%	24.1%	16.7%	13.9%

keeping the multi-patch MLLM fixed. Table 9 reports the benchmark accuracy, and Table 10 reports the average computational cost and routing distribution.

DINOv3-Base achieves the best overall accuracy and efficiency. Although DINOv3-Small is lighter, it selects p=8 more frequently, producing more visual tokens and higher downstream computation. The comparable results across all encoders also demonstrate that the routing framework is robust to the choice of feature extractor, which introduces only a negligible parameter overhead.

A.5.4 Benchmark-level Efficiency Analysis

The candidate set $\mathcal{P} = \{8, 10, 12, 14\}$ spans a wide range of token counts, so the efficiency of MoP depends on its routing decisions. To verify that the reported FLOPs reduction is not an average that hides dataset-specific overhead, we report per-benchmark FLOPs, inference latency, and GPU memory for each fixed patch size and for MoP in Tables 11 and 12. The efficiency gain is measured against p=8, which generally provides the strongest fixed-patch performance.

Compared with always selecting p=8, MoP reduces FLOPs by 1.34×–1.52× and latency by 1.09×–1.22× across all benchmarks, while preserving or improving accuracy. The gain is consistent across benchmarks rather than being caused by a specific routing preference; even on RealWorldQA, where the router favors p=8, MoP still saves 1.35× FLOPs. The lightweight router introduces a limited additional memory overhead of about 3–4%. Together with the accuracy results, this shows that MoP moves the operating point beyond the fixed patch-size Pareto curve: it achieves better performance than individual fixed patch choices while requiring lower computation than always using the finest patch size.

A.6 Formal Description of Patch Embedding Transfer

Notation

Let $\mathcal{X} \subset \mathbb{R}^d$ denote the distribution of input image patches, and each patch is represented as a vector $x \in \mathcal{X}$. We summarize all symbols used in this section as follows:

- $W \in \mathbb{R}^{D\times(C*P*P)}$: original patch embedding weight.
- $\hat{W} \in \mathbb{R}^{D\times(C*\hat{P}*\hat{P})}$: refined patch embedding weight after patch-size transformation.
- $B \in \mathbb{R}^{(C*P*P)\times(C*\hat{P}*\hat{P})}$: linear mapping that aligns a resized patch to the original embedding space.
- $\Sigma = \mathbb{E}_{x\sim\mathcal{X}}[xx^T]$: (uncentered) covariance matrix of the input distribution.
- $\langle\cdot,\cdot\rangle$: inner product between a patch vector and a patch embedding weight.
- $||v||_\Sigma^2 = v^T\Sigma v$: Σ-induced Mahalanobis norm.
- $(\cdot)^+$: Moore–Penrose pseudoinverse operator.

Table 11: **Per-benchmark FLOPs and inference latency** of fixed patch sizes and MoP on the Qwen3-VL-4B backbone. The ratio columns report the p=8 value divided by the MoP value.

	FLOPs (TFLOPs)						Latency (ms)					
Benchmark	p=8	p=10	p=12	p=14	MoP	Ratio	p=8	p=10	p=12	p=14	MoP	Ratio
AI2D	12.4	8.1	5.8	4.4	8.5	1.46×	191.0	157.2	130.4	122.1	168.1	1.14×
ChartQA	19.1	12.4	8.7	6.7	13.6	1.40×	365.9	309.5	257.2	233.6	308.9	1.18×
MMBench	7.7	5.2	3.8	2.9	5.2	1.48×	133.8	122.2	114.3	106.8	119.2	1.12×
MMStar	9.0	6.0	4.3	3.3	6.2	1.45×	149.7	127.3	122.7	108.0	129.7	1.15×
POPE	11.8	7.6	5.5	4.0	8.1	1.46×	163.4	131.7	118.0	118.6	142.1	1.15×
RealWorldQA	24.6	16.0	11.3	8.3	18.2	1.35×	330.6	242.2	188.4	166.4	290.7	1.14×
SQA	8.2	5.5	4.1	3.2	6.1	1.34×	151.2	126.7	110.1	108.2	127.1	1.19×
SEED-Img	8.4	5.7	4.0	3.1	5.7	1.47×	129.7	113.5	111.9	103.2	119.1	1.09×
TextVQA	25.2	16.1	11.4	8.4	16.6	1.52×	392.2	307.9	255.9	236.1	321.7	1.22×

Table 12: **Per-benchmark peak GPU memory (MB)** of fixed patch sizes and MoP. The overhead of MoP comes from the lightweight router.

Benchmark	p=8	p=10	p=12	p=14	MoP
AI2D	9052.7	8877.7	8784.4	8725.5	9361.7
ChartQA	9327.1	9059.2	8916.4	8826.9	9574.3
MMBench	8864.8	8761.8	8696.4	8662.6	9229.2
MMStar	8915.2	8794.9	8721.0	8680.4	9280.8
POPE	9030.3	8858.8	8778.5	8702.6	9352.1
RealWorldQA	9544.0	9208.6	9026.5	8905.5	9754.9
SQA	8882.7	8773.4	8714.9	8674.9	9263.9
SEED-Img	8895.3	8783.6	8702.7	8665.2	9250.8
TextVQA	9569.7	9204.9	9029.1	8912.8	9693.7

Derivation

We borrow the PI-resize(Pseudo-Inverse resize) derivation adapted from (Beyer et al., 2023) to conduct Patch Embedding Transfer.

Our objective is to obtain a new patch embedding weight $\hat{W}$ such that the discrepancy between the transformed embedding and the original one is minimized. This constraint ensures that the structural adjustment of the patch embedding layer does not lead to information loss. The formula is as follows:

$$\begin{aligned}
&\mathbb{E}_{x\sim\mathcal{X}}\left[(\langle x, W\rangle - \langle Bx, \hat{W}\rangle)^2\right] \\
=&\mathbb{E}_{x\sim\mathcal{X}}\left[(x^T(W - B^T\hat{W}))^2\right] \\
=&\mathbb{E}_{x\sim\mathcal{X}}\left[((W - B^T\hat{W})^T x)(x^T(W - B^T\hat{W}))\right] \\
=&(W - B^T\hat{W})^T\mathbb{E}_{x\sim\mathcal{X}}\left[xx^T\right](W - B^T\hat{W}) \\
=&||W - B^T\hat{W}||^2_\Sigma,
\end{aligned} \tag{7}$$

where $||v||^2_\Sigma = v^T\Sigma v$ and $\Sigma = \mathbb{E}_{x\sim\mathcal{X}}xx^T$ is the (uncentered) covariance matrix of $\mathcal{X}$. When $\mathcal{X} = \mathcal{N}(0, I)$, this recovers the standard Euclidean norm $||w - B^T w_{new}||^2$. Next, recall that the pseudoinverse yields the least-squares solution for a linear system of equations:

$$(B^T)^+ W \in \arg\min_{\hat{W}} ||W - B^T\hat{W}||^2. \tag{8}$$

We can also derive an analytic solution for an arbitrary covariance matrix $\Sigma = \mathbb{E}_{x\sim\mathcal{X}}xx^T$. Noting that $||v||^2_\Sigma = v^T\Sigma v = (\sqrt{\Sigma}v)^T(\sqrt{\Sigma}v) = ||\sqrt{\Sigma}v||^2$, we can rewrite the objective as:

$$||W - B^T\hat{W}||^2_\Sigma = ||\sqrt{\Sigma}W - \sqrt{\Sigma}B^T\hat{W}||^2. \tag{9}$$

The optimal solution is then given by:

$$(\sqrt{\Sigma}B^T)^+\sqrt{\Sigma}W \in \arg\min_{\hat{W}} ||W - B^T\hat{W}||^2. \tag{10}$$

A.7 Future Work and Limitation

A.7.1 Limitations

MoP still has several limitations. First, the candidate patch sizes are predefined during multi-patch adaptation. Although these candidates are selected based on our preliminary analysis of the favorable granularity range, the tokenization space remains discrete and is not fully flexible. Second, the current router is trained using pre-collected per-patch forward losses as supervision. This loss-proxy-based and decoupled optimization can learn useful patch-size preferences for different inputs, but it may not fully exploit the rich semantic interactions inside the VLM.

A.7.2 Future Work

In future work, we plan to explore online and lossless tokenizer conversion methods to support more flexible patch-size adaptation. Beyond image-level routing, we are also interested in region-adaptive tokenization, where different image regions can be encoded with different patch sizes and composed within a unified visual representation. Finally, we aim to integrate the router and multi-patch tokenizer into an end-to-end trainable architecture, leading to a more native and expressive MoP framework.

A.8 Case Study

Case Study of MoP Routing Decisions



Input image

Question: How many seconds left to cross the street for a pedestrian?

Options: A. 10 B. 11 C. 12

Multi-patch (p=8) **answer(✓):** B.

Multi-patch (p=10) **answer(✗):** A.

Multi-patch (p=12) **answer(✗):** A.

Multi-patch (p=14) **answer(✗):** A.

MoP selects(✓): $\mathbf{p{=}8}$.



Input image

Question: Where is the elephant relative to the bird?

Options: A. The elephant is above the bird. B. The elephant is to the right of the bird. C. The elephant is below the bird.

Multi-patch (p=8) **answer(✗):** B.

Multi-patch (p=10) **answer(✓):** C.

Multi-patch (p=12) **answer(✓):** C.

Multi-patch (p=14) **answer(✓):** C.

MoP selects(✓): $\mathbf{p{=}14}$.