A. Additional Details

A.1. Prompt Details

AdaptVision utilizes three types of prompts. First, to equip the VLM with basic tool-using capability, we follow the Qwen2.5-VL cookbook [3] to design prompts for the bounding box tool (Table 5). Second, since VQA tasks are typically diverse and open-ended, we adopt an LLM-as-judge approach to evaluate answer correctness. As shown in Table 6, following Yang et al. [46], we design a judging prompt for GPT-4o to produce binary evaluations (1 for correct, 0 for incorrect). Third, to encourage efficient tool exploration, we prompt GPT-4o to evaluate the relevance of cropped regions, producing a binary reward for region correctness (Table 7).

A.2. Training and Evaluation Details

AdaptVision is based on Qwen2.5-VL-7B-Instruct [3]. We employ veRL [33] framework for RL training. During training, we set the batch size as 512 with mixed-precision (FP16) training. The mini-batch size is 32. We drop the KL term during policy optimization. For each prompt, we sample 16 candidate responses (i.e., $G = 16$) using a temperature of 1.0. The upper and lower clip ratios are 0.24 and 0.20, respectively. We set the maximum prompt length and the maximum response length as 8192. All experiments were conducted on 4 nodes, each with 8 H20 GPUs. The model was trained for 80 steps, using the AdamW optimizer with a learning rate of $1e{-}6$, $\beta = (0.9, 0.999)$, and a weight decay of 0.01. During inference, we use the vLLM framework and set the temperature to 0.

A.3. Additional Results

We further compare AdaptVision with previous efficient VLM methods with different visual token retention ratios. As shown in Table 4, while the performance of FastV, SparseVLM, and VisionZip degrades with reduced token ratios, AdaptVision maintains superior performance with significantly fewer visual tokens.

B. Qualitative Results

We provide further case studies to illustrate AdaptVision's adaptive token usage. As shown in Fig. 8, in scenarios where a low-resolution image provides enough information, AdaptVision correctly chooses to answer directly—matching the behavior of the Qwen2.5-VL Down-sample model. Conversely, in cases where detailed visual information is essential (Fig. 9), the Down-sample model often fails due to recognition errors caused by insufficient resolution (e.g., misreading "15" as "75"). Under the same conditions, AdaptVision actively invokes the bounding box tool, accurately localizes informative regions, and produces correct answers with only a marginal increase in visual token consumption relative to the Down-sample model. These examples validate AdaptVision's ability in coarse-to-fine visual reasoning and its capacity to autonomously tailor visual token usage to each input.

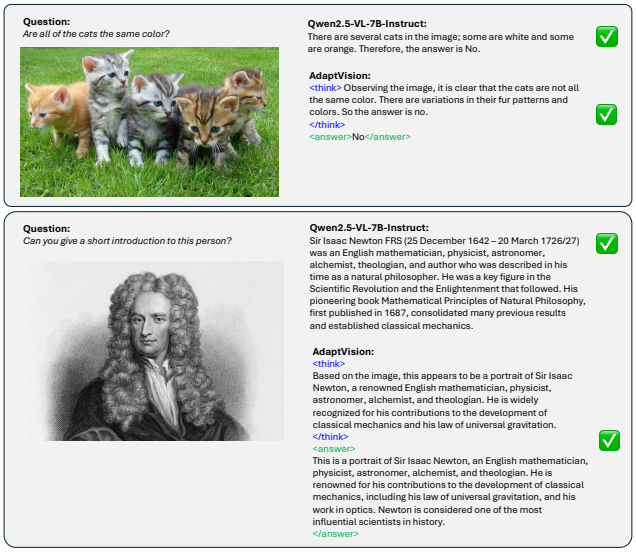

Figure 8. **Case of direct answer in AdaptVision.**

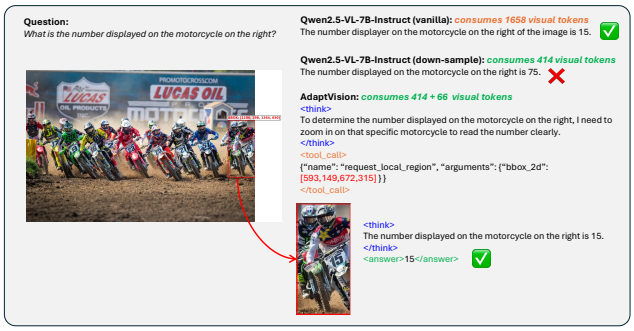

Figure 9. **Case of tool call in AdaptVision.**

Table 2. Sensitivity analysis on λ and α.

	$\lambda = 0.2$	$\lambda = 0.25$	$\lambda = 0.3$	$\lambda = 0.35$	$\lambda = 0.4$	$\alpha = 1$	$\alpha = 2$
RealWorldQA	64.81	66.27	67.32	66.84	66.43	66.91	67.32
MME	2368	2398	2379	2378	2394	2400	2379

Table 3. Comparison of different reward models.

Model	RealWorldQA	POPE	MME	MathVista	MMVet	#Token↓
VisionThink	65.6	86.3	2320	62.2	61.7	51.86%
AdaptVision (GPT-4o)	67.32	86.8	2379	65.9	64.8	30.66%
AdaptVision (Qwen3VL)	66.47	86.8	2313	64.7	62.5	34.18%

C. More Discussion on DTPO

To help readers to better understand DTPO, we summarize the core contributions of DTPO. First, DTPO decouples advantage estimation by computing distinct advantages for tool and outcome rewards, thereby preventing different rewards from interfering with each other and enabling accurate advantage estimation. By assigning different advantages to distinct tokens, it achieves more precise credit assignment. Second, DTPO decouples policy loss by turns and normalizes the contributions of tool and answer tokens separately, ensuring balanced optimization across tokens with distinct functions. The core contribution of DTPO lies in 1) better credit assignment across different turns and 2) resolving the imbalanced optimization problem across multiple turns. Consequently, DTPO is not limited to the visual reasoning scenario and can be adapted to other multi-turn RL scenarios.

D. Robustness of DTPO

The performance of models trained with different λ and α are reported in Table 2. The results show that DTPO is robust to hyperparameters.

E. Reward Models for AdaptVision

AdaptVision uses GPT-4o to compute the R_{crop} reward. Here we investigate whether this R_{crop} reward can be computed using a smaller, open-source VLM instead of GPT-4o. We conducted an experiment using a smaller open-source VLM, Qwen3-VL-4B-Instruct, to replace GPT-4o as the judge model. From Table 3, we found that the model still achieves competitive performance and outperforms VisionThink.

F. Generalize with Other VLM Architectures

Our method is architecture-agnostic. It modifies the model's interaction logic and training methodology rather than the model backbone. Thus, it can be applied to other VLM architectures such as Qwen3-VL or InternVL. We chose Qwen2.5-VL-7B-Instruct as our base model because it is a popular and strong open-source VLM baseline at this time. The modular design of AdaptVision ensures its broad applicability, and we encourage future research to extend AdaptVision to alternative VLM architectures.

G. Extension to Multi-round Tool Calls

Since our primary goal is to build an efficient VLM, we limit the interaction turns to minimize latency. While we believe that multi-turn tool-use could further improve accuracy by obtaining more fine-grained visual information, it comes at the cost of increased inference time. Nevertheless, we encourage future research to extend AdaptVision to multi-round tool calls to improve accuracy while preserving inference efficiency. For example, applying a deblurring module after zooming in on small objects can significantly improve clarity. This may improve performance in certain cases.

Table 4. **Performance comparison with previous efficient VLM methods.** Vanilla denotes the Qwen2.5-VL-7B-Instruct model. Down-Sample uses a 1/4-resolution image as input to the Vanilla model. "#Token" indicates the visual token consumption ratio relative to the vanilla model across all benchmarks. "Avg." denotes the average performance relative to the vanilla model on all benchmarks. "Method (xx%)" denotes static methods retaining xx% visual tokens.

Method	**ChartQA** test	**OCRBench** test	**DocVQA** val	**MME** test	**MMVet** test	**RealWorldQA** test	**POPE** test	**MathVista** testmini	**MathVerse** testmini	**#Token**↓	**Avg.**↑
Retain 100% Visual Tokens Across All Benchmarks											
Vanilla	79.8 100%	81.5 100%	95.1 100%	2316 100%	61.6 100%	68.6 100%	86.7 100%	68.2 100%	46.3 100%	100%	100%
Retain 25% Visual Tokens Across All Benchmarks											
Down-Sample	62.9 78.8%	68.8 84.4%	94.3 99.1%	2270 98.0%	54.5 88.5%	68.8 100.3%	82.8 95.5%	62.2 91.2%	43.1 93.1%	25%	92.1%
Retain 50% Visual Tokens Across All Benchmarks											
SparseVLM (50%)	73.2 91.7%	75.6 92.7%	66.8 70.2%	2282 98.5%	51.5 83.6%	68.4 99.7%	85.5 98.6%	66.6 97.6%	45.1 97.4%	50%	92.2%
FastV (50%)	72.6 91.0%	75.8 93.0%	93.6 98.4%	2308 99.6%	52.8 85.7%	68.8 100.3%	84.7 97.7%	63.7 93.4%	45.0 97.2%	50%	95.8%
VisionZip (50%)	71.5 89.6%	70.5 86.5%	93.8 98.6%	2209 95.4%	57.0 92.5%	68.6 100%	86.3 99.5%	64.1 93.9%	45.1 97.4%	50%	94.8%
Retain 70% Visual Tokens Across All Benchmarks											
SparseVLM (70%)	75.8 94.9%	79.3 97.3%	68.7 72.2%	2276 98.3%	53.7 87.2%	68.5 99.8%	85.4 98.5%	66.3 97.2%	45.1 97.4%	70%	93.6%
FastV (70%)	71.2 96.7%	82.2 100.8%	94.4 99.3%	2342 101.1%	56.0 90.9%	68.6 100%	85.9 99.1%	65.9 96.6%	46.9 101.3%	70%	98.4%
VisionZip (70%)	76.8 96.2%	80.9 99.3%	94.5 99.4%	2334 100.8%	60.0 97.4%	68.2 99.4%	86.4 99.7%	68.9 101.0%	45.8 98.9%	70%	99.1%
Dynamic Methods											
VisionThink	73.6 92.2%	76.8 94.2%	92.9 97.7%	2320 100.2%	61.7 100.2%	65.6 95.6%	86.3 99.5%	62.2 91.2%	42.5 91.8%	52%	95.8%
VisionThink†	73.88 92.6%	80.8 99.1%	93.7 98.5%	2392 103.3%	60.18 97.7%	68.37 99.7%	86.69 100.0%	65.7 96.3%	45.68 98.7%	99%	98.4%
AdaptVision	75.92 95.1%	76.9 94.4%	92.6 97.4%	2379 102.7%	64.8 105.2%	67.32 98.1%	86.8 100.1%	65.9 96.6%	42.3 91.4%	**33%**	**97.9%**

Table 5. **Prompt Template for adaptively visual acquisition.** Question will be replaced with the specific question during training and inference.

SYSTEM PROMPT:

You are a helpful assistant.

Tools

You may call the function tool shown below to assist with the user query.

You are provided with the function signature within <tools></tools> XML tags:

```
<tools>
{
  "type": "function",
  "function":{
    "name_for_human": "request_local_region",
    "name": "request_local_region",
    "description": "Request a high-resolution local region of the current image and zoom in",
      "parameters": {
      "properties": {
        "bbox_2d": {
          "type": "array",
          "items": {
            "type": "integer"
          }
          "minItems": 4,
          "maxItems": 4,
          "description":The bounding box of the region to crop, as [x1, y1, x2, y2], where (x1, y1) is the top-left corner of the target region and (x2, y2) is the bottom-right corner of the target region. The bounding box should be in the absolute pixel coordinates of the current image.",
        }
      }
      "required": ["bbox_2d"],
      "type": "object",
    },
  "args_format": "Format the arguments as a JSON object."
  }
}
</tools>
```

For each function call, return a json object with the function name and the corresponding argument within <tool_call></tool_call> XML tags:

```
<tool_call>
{"name": <function-name>, "arguments":<args-json-object>}
</tool_call>
```

USER PROMPT:

Answer the question based on the image provided. You must conduct reasoning within <think> and </think> first in each of your reasoning steps. You may call ONE function tool per step to help you better solve the problem. Place the function tool within <tool_call> and </tool_call> at the end of each step to perform a function call. You should continue your reasoning process within <think> and </think> based on the content returned by the function tool. Once you confirm your final answer, place the final answer inside <answer> and </answer>. For mathematical or multiple-choice problem, wrap the answer value or choice with \boxed{}. Here is the image and question: Question.

Table 6. **Prompt Template for LLM as Final Answer Judge.** Question, Ground Truth and Prediction are dynamically replaced with the specific question, ground truth and model prediction during evaluation.

SYSTEM PROMPT:

You are an intelligent chatbot designed for evaluating the correctness of generative outputs for question-answer pairs.
Your task is to compare the predicted answer with the correct answer and determine if they match meaningfully. Here's how you can accomplish the task:
INSTRUCTIONS:
- Focus on the meaningful match between the predicted answer and the correct answer.
- Consider synonyms or paraphrases as valid matches.
- Evaluate the correctness of the prediction compared to the answer.

USER PROMPT:

I will give you a question related to an image and the following text as inputs:
1. **Question Related to the Image**: Question
2. **Ground Truth Answer**: Ground Truth
3. **Model Predicted Answer**: Prediction
Your task is to evaluate the model's predicted answer against the ground truth answer, based on the context provided by the question related to the image. Consider the following criteria for evaluation:
- **Relevance**: Does the predicted answer directly address the question posed, considering the information provided by the given question?
- **Accuracy**: Compare the predicted answer to the ground truth answer. You need to evaluate from the following two perspectives:
(1) If the ground truth answer is open-ended, consider whether the prediction accurately reflects the information given in the ground truth without introducing factual inaccuracies. If it does, the prediction should be considered correct.
(2) If the ground truth answer is a definitive answer, strictly compare the model's prediction to the actual answer. Pay attention to unit conversions such as length and angle, etc. As long as the results are consistent, the model's prediction should be deemed correct.
Output Format:
Your response should include an integer score indicating the correctness of the prediction: 1 for correct and 0 for incorrect. Note that 1 means the model's prediction strictly aligns with the ground truth, while 0 means it does not.
The format should be Score: 0 or 1

Table 7. **Prompt Template for Judging the Correctness of Bounding Box.** Question are dynamically replaced with the specific question during evaluation.

SYSTEM PROMPT: **Your Role:** You are an AI agent that identifies relevant visual evidence. **Your Goal:** Determine if an image CROP contains the **primary subject** of a given question. **Your Golden Rule:** Your main task is to check for **presence**, not completeness. As long as the main object or area the question is asking about is clearly visible in the crop, it is considered relevant. **Criteria for 'Score: 0' (Strictly Enforced):** - The core subject of the question is completely absent from the image. - The image is so blurry or corrupted that the subject is **unrecognizable**. - The image shows something completely unrelated (e.g. question is about a car, image shows a tree). **Your Task:** Now, analyze the user-provided image and question following this exact process. Your response MUST only contain 'Score: 1' or 'Score: 0'.
USER PROMPT: Given a question and a cropped image region, answer with 'Score: 1' if the cropped region provide information to answer the question, otherwise answer 'Score: 0'. Question: Question."