MEDCODE: MEDICAL VISUAL REASONING BEYOND PREDEFINED TOOLKITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Medical vision language models (VLM) have benefited from explicit reasoning, but most existing approaches reason over a visual representation that is encoded only once. Recent tool augmented methods allow models to revisit the image, yet their analyses remain limited to procedures defined in advance. In this paper, we present MedCode, a framework that enables medical VLMs to construct question specific image analyses through executable code. Instead of selecting from a predefined toolkit, MedCode writes programs using general image processing and numerical operations. We further develop MedCodeLab, a shared general purpose environment that executes these programs and returns textual results and derived images for subsequent reasoning. To teach this capability, we construct MedCode-50k, which contains 49,817 verified multi turn trajectories across eight imaging modalities and four question categories. We train an 8B model through trajectory based supervised fine tuning followed by agentic reinforcement learning on questions selected according to rollout difficulty. On the in distribution test set, MedCode raises the accuracy of its backbone from 75.2% to 82.8%. It also generalizes to four unseen data sources, where it improves average accuracy from 76.2% to 85.5%. MedCode consistently outperforms crop based reasoning and predefined image operations, demonstrating the value of constructing flexible analyses for medical visual reasoning.

1 INTRODUCTION

Vision-language models have made rapid progress in medical image understanding (Moor et al., 2023; Luo et al., 2023; Chen et al., 2024a; Tu et al., 2024), with promising results in visual question answering, disease diagnosis, and report generation (Li et al., 2023a; Chen et al., 2024b; Pellegrini et al., 2023). Recent studies suggest that these models benefit from reasoning over image evidence before producing a final answer. In particular, chain-of-thought supervision and reinforcement learning with verifiable rewards (Pan et al., 2025; Lai et al., 2026) have encouraged medical VLMs to produce more structured reasoning instead of direct predictions (Wei et al., 2022; Sun et al., 2026; Huang et al., 2025; Hsieh et al., 2023). This can improve diagnostic accuracy and make model decisions easier to examine, which is important in clinical applications.

Despite this progress, reasoning in most medical VLMs remains text based. The image is typically encoded once, after which the model relies on a fixed visual representation throughout the reasoning process (Zhang et al., 2023). It may describe visible findings, but it cannot directly return to a suspicious region or measure an anatomical structure. This limitation becomes particularly important when an answer depends on subtle or quantitative evidence, such as lesion extent, intensity contrast with surrounding tissue, or boundary morphology (Van Griethuysen et al., 2017). It may also increase the risk of visual hallucination (Li et al., 2023b; Guan et al., 2024; Leng et al., 2024). Recent studies have begun to address this problem by allowing models to invoke image-analysis tools during reasoning (Wu et al., 2023; Surís et al., 2023; Gupta & Kembhavi, 2023). Medical agents such as MMedAgent (Li et al., 2024) and AURA (Fathi et al., 2025) equip VLMs with specialist models for tasks such as segmentation (Ma et al., 2024) and detection (Baumgartner et al., 2021). Ophiuchus (Jiang et al.) and MedVistaGym (Lu et al., 2026) further train medical VLMs to interleave reasoning with operations such as cropping and zooming, then use the resulting observations in subsequent reasoning.

Although these methods differ in how tools are selected and invoked, they generally provide the model with a predefined toolkit. This design introduces two challenges. First, a predefined toolkit

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.

constrains the analyses that the model can perform. The model may select a tool, adjust its parameters, and combine several calls (Yao et al., 2022; Schick et al., 2023), but its analytical space remains bounded by the procedures implemented in advance (Wang et al., 2024). Some questions instead require a question-specific computation, such as comparing two measured extents or tracing an intensity profile across a lesion boundary. Such procedures cannot always be recovered by rearranging existing tools. Second, the extensibility of these approaches to different modalities and tasks is limited. Medical toolkits commonly depend on specialist models trained for particular data distributions (Isensee et al., 2021). A detector developed for chest CT, for example, provides little useful support for a fundus photograph. Broader coverage therefore depends on adapting existing components or integrating new ones for each modality and task.

To address these challenges, we propose MedCode, a framework that enables medical VLMs to construct question-specific image analyses through executable code. Instead of selecting from a predefined collection of medical tools, MedCode writes the required analysis using general-purpose code like image-processing and numerical operations. Each program determines which operations to run and how they should be parameterized. Intermediate computations and control flow are defined within the same program. This flexibility allows the model to construct analyses that are not covered by a fixed set of tool interfaces. We further develop MedCodeLab as a shared general-purpose environment for executing these programs. MedCodeLab returns textual results and derived images from actual execution, allowing the model to use this evidence in subsequent reasoning. Because the same environment is used across all modalities and tasks, MedCode does not require a separate toolkit or specialist perception model for each one.

To teach the model to generate these programs and reason from execution feedback, we build MedCode-50k, a code-augmented medical reasoning dataset comprising 49,817 multi-turn trajectories that interleave reasoning, executable programs, and execution feedback. The dataset spans eight imaging modalities and four medical question categories, providing broad coverage of diverse imaging settings and analytical requirements. Based on MedCode-50k, we perform supervised fine-tuning on Qwen3-VL-8B (Bai et al., 2025a), teaching the model to generate executable analyses and use the returned evidence. We then select 16k questions through difficulty-aware selection for agentic reinforcement learning with MedCodeLab, further improving its ability to construct effective analyses for challenging questions. We evaluate MedCode on a diverse set of medical image reasoning tasks. It improves over text-based reasoning by 10.1% and crop-and-resize reasoning by 6.8%. It also exceeds the predefined tool-use baseline by 3.6% without relying on specialist perception models.

Our contributions are summarized as follows:

- We propose MedCode, a framework that enables medical VLMs to construct question-specific image analyses through executable code. This design extends medical visual reasoning beyond the capabilities of predefined tool interfaces.
- We develop MedCodeLab, a shared general-purpose environment that executes model-generated programs and returns their outputs for subsequent reasoning, supporting different modalities and tasks.
- We construct MedCode-50k, a code-augmented medical reasoning dataset containing 49,817 multi-turn trajectories across eight imaging modalities and four question categories.
- Comprehensive experiments show that MedCode outperforms text-based reasoning, single-crop reasoning, and predefined tool-use methods. Further ablations confirm the importance of execution-trajectory training.

2 METHOD

2.1 OVERVIEW

MedCode enables medical VLMs to construct question-specific image analyses through executable code, as illustrated in Figure 1. To support program execution and iterative reasoning, we develop **MedCodeLab**, a shared environment that provides general-purpose image-processing and numerical operations and returns execution results for subsequent reasoning. Using this environment, we collect and verify multi-turn trajectories across medical tasks and imaging modalities to construct

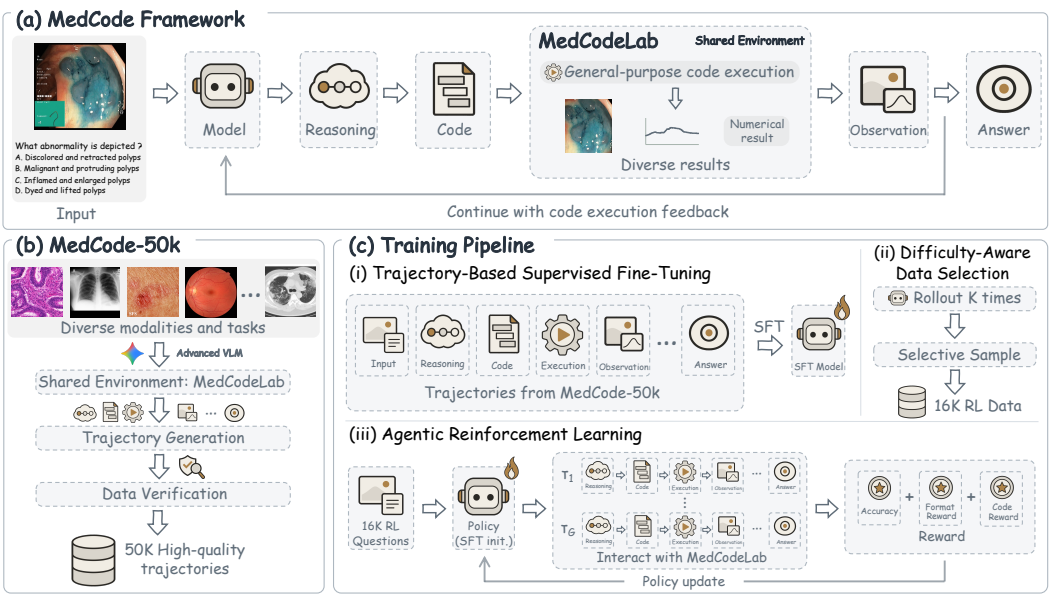


Figure 1: Overview of MedCode. (a) The model generates question-specific code and uses execution feedback from MedCodeLab to guide medical visual reasoning. (b) Verified teacher trajectories collected in the same environment form MedCode-50k. (c) Trajectory-based supervised fine-tuning is followed by reinforcement learning on 16k questions selected through difficulty assessment. Rewards account for answer correctness, response format, and code execution, with code credit granted only when the final answer is correct.

MedCode-50k. We then train **MedCode** through supervised fine-tuning on these trajectories, followed by reinforcement learning on questions selected through rollout-based difficulty assessment.

Formally, given an input $x = (I, q, \mathcal{C})$, where I is a medical image, q is a question, and $\mathcal{C}$ is the set of candidate answers, a vision-language policy π_θ interacts with MedCodeLab over multiple turns. At each nonterminal turn t, the model produces a response u_t containing a rationale and a program c_t. MedCodeLab executes the program and returns an observation o_t, which may contain textual results or derived images. The interaction follows

$$o_t = \mathcal{E}(I, c_t), \qquad h_{t+1} = h_t \oplus u_t \oplus o_t, \tag{1}$$

where $\mathcal{E}$ denotes the program execution function provided by MedCodeLab, h_t is the multimodal interaction history available at turn t, and $\oplus$ denotes appending new information to this history. The interaction is initialized with $h_1 = x$. The model continues this interaction until it produces a final response u_T. The resulting trajectory is

$$\tau = (u_1, o_1, \ldots, u_{T-1}, o_{T-1}, u_T). \tag{2}$$

The reference answer y is withheld from the policy input and is used only for trajectory filtering and training rewards.

2.2 MedCodeLab

General Code Environment. MedCodeLab implements the execution function $\mathcal{E}$ in Eq. 1 and provides the shared interface used for data construction, model training, and inference. It exposes a single tool named `code_interpreter`, which runs generated code in a general-purpose Python code environment. The environment provides standard support for image processing, numerical computation, and scientific visualization. The model specifies each analytical procedure directly in code, including its operations, parameters, intermediate computations, and control logic. New procedures can therefore be constructed without extending the environment with separately packaged tools.

Stable Execution and Multimodal Feedback. The model invokes MedCodeLab by placing the generated code within `<tool_call>...</tool_call>` tags. MedCodeLab extracts the code

and executes it in an isolated subprocess within a temporary directory created for that call. The input image is assigned a randomized filename and made available through `IMAGE_PATH`. Files produced during execution are written to `OUTPUT_DIR`. Randomizing the filename prevents information in the original file path from providing unintended cues about the image or its source. Runtime state is not preserved between calls. Instead, the returned observations are appended to the multimodal conversation and provide the context for subsequent reasoning. Code operates on the source-resolution image, while returned images are processed by the image processor of the VLM before being added to the conversation.

MedCodeLab records textual output, error messages, generated images, and the execution status. The observation returned at turn t is represented as

$$o_t = \left(z_t^{\text{out}}, z_t^{\text{err}}, \mathcal{V}_t, \sigma_t\right), \tag{3}$$

where z_t^{out} and z_t^{err} denote the textual output and error information, $\mathcal{V}_t$ is the set of generated images, and σ_t records the execution status. Figures and image files created during execution are collected automatically. Once the model emits a tool call, any assistant content following the call in the same response is ignored. MedCodeLab places the resulting observation within `<tool_response>...</tool_response>` tags and appends it to the multimodal context before model generation resumes. The model therefore receives feedback produced by actual execution rather than feedback synthesized in its own response. Each execution has a time limit of 60 seconds. Execution failures are returned through the same response format, allowing the model to revise its code in subsequent turns.

2.3 MEDCODE-50K

Diverse Source Data and Partitioning. We construct MedCode-50k from the open-access subset of OmniMedVQA (Hu et al., 2024). After removing modality-recognition questions, the source pool contains 77,431 question-answer pairs from 40 datasets. It covers eight imaging modalities, including X-ray, CT, MRI, ultrasound, OCT, fundus photography, dermoscopy, and microscopy. The questions fall into four categories covering anatomical identification, disease diagnosis, lesion grading, and other biological attributes. We partition the source data before trajectory generation and filtering. Questions from the development sources are divided into training, validation, and in-distribution test sets using an 80/10/10 split, yielding 59,904, 7,659, and 7,747 examples, respectively. To evaluate generalization beyond the training distribution, we additionally reserve BreakHis (Spanhol et al., 2015), PAD-UFES-20 (Pacheco et al., 2020), PALM2019 (Fang et al., 2024), and Chest X-Ray PA (Asraf & Islam, 2021) as a source-held-out test set containing 2,121 examples.

Multi-turn Trajectory Generation. We use Gemini 3.1 Pro as the teacher model. The teacher receives the image, question, answer options, and instructions for interacting with MedCodeLab. The reference answer is not included in its input. During generation, the teacher produces a reasoning step and executable code, observes the output returned by MedCodeLab, and continues until it reaches a final answer. We record the complete multimodal trajectory, including the source identifier, assistant responses, generated code, execution results, derived images, and final answer. We do not provide target code or prescribe an analysis procedure for any modality or question category.

Answer Verification and Quality Filtering. We first retain trajectories whose final answers match the reference options. We then remove trajectories in which every execution fails, the generated code does not perform image or numerical analysis, or the interaction format cannot be parsed. Code that only prints predetermined text, assigns constant values, or executes `pass` is treated as an invalid analysis. Printing values obtained from an actual computation is allowed. A trajectory may contain an intermediate execution failure if the teacher subsequently corrects the code and completes the analysis successfully. Every retained trajectory contains at least one successful execution and terminates with a valid final answer. After filtering, 49,817 training questions have an accepted trajectory. These trajectories form MedCode-50k. Each example contains execution feedback followed by further reasoning or a final response, rather than pairing the image with a standalone textual explanation.

2.4 TRAINING MEDCODE

We train MedCode in two stages. Supervised fine-tuning first establishes the ability to generate executable analyses and reason from their outputs. We then use rollouts from the resulting policy to

identify questions suitable for reinforcement learning and optimize the model through interactions with MedCodeLab.

Trajectory-Based Supervised Fine-Tuning. We initialize MedCode from Qwen3-VL-8B-Instruct and finetune it on MedCode-50k. Each teacher trajectory is serialized using the native multimodal conversation template. The original image, user question, and execution observations are provided as context, while the training loss is applied only to assistant-generated tokens. These tokens include the reasoning, generated code, tool-call delimiters, and final answer. Outputs returned by MedCodeLab are therefore used as conditioning information rather than prediction targets. For a trajectory τ, let $\mathcal{A}(\tau)$ denote the positions of assistant-generated text tokens, z_j denote the token at position j, and h_j denote its full text-and-image prefix. The token-normalized SFT objective for a batch $\mathcal{B}$ is

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\frac{\sum_{\tau\in\mathcal{B}}\sum_{j\in\mathcal{A}(\tau)} \log \pi_\theta(z_j \mid h_j)}{\sum_{\tau\in\mathcal{B}} |\mathcal{A}(\tau)|}. \tag{4}$$

This objective supervises both code generation and reasoning conditioned on execution feedback. No gradient is propagated through code execution.

Difficulty-Aware Data Selection. After supervised fine-tuning, we evaluate the resulting policy on all questions in MedCode-50k. For each question x, we sample $K = 4$ independent trajectories and count the number of correct final answers

$$k(x) = \sum_{j=1}^{K} \mathbf{1}\left[\widehat{y}^{(j)} = y\right], \tag{5}$$

where $\widehat{y}^{(j)}$ denotes the answer predicted in the j-th rollout, and $\mathbf{1}[\cdot]$ is the indicator function. We retain about 15k questions with $1 \le k(x) \le 3$. These questions can be solved by the current policy but still leave room for improvement through online exploration (Yu et al., 2026). Questions with $k(x) = 0$ are excluded because none of the sampled trajectories provides a successful outcome for comparison. We also sample 1,000 distinct questions with $k(x) = 4$ to preserve capabilities already acquired during supervised fine-tuning. The resulting reinforcement learning set contains 16k questions.

Agentic Reinforcement Learning. Using the data selected above, we further optimize the SFT model with Group Relative Policy Optimization (GRPO). Unlike supervised fine-tuning on fixed teacher trajectories, this stage allows the policy to generate new programs, observe their execution results, and continue reasoning from the returned feedback. At each update, the rollout policy π_{old} samples a group of $G = 8$ trajectories for each question. Each trajectory interacts with MedCodeLab independently and receives its own textual and visual observations. These trajectories are subsequently evaluated and used to update the policy.

Trajectory Reward. We evaluate each sampled trajectory according to its final answer, format, and program execution. The trajectory-level reward is

$$R(\tau) = r_{\mathrm{acc}}(\tau) + 0.5\, r_{\mathrm{fmt}}(\tau) + 0.5\, r_{\mathrm{acc}}(\tau) r_{\mathrm{code}}(\tau). \tag{6}$$

The answer reward r_{acc} is one when the trajectory produces a uniquely parseable final option that matches the reference answer, and is zero otherwise. The format reward r_{fmt} is one when the trajectory follows the tool-call protocol and terminates with a single valid final answer tag. The code reward r_{code} is one when at least one nontrivial program executes successfully and produces a nonempty textual or visual observation. The product $r_{\mathrm{acc}}(\tau) r_{\mathrm{code}}(\tau)$ ensures that successful code execution receives a reward only when the final answer is correct. The code reward is assigned once per trajectory regardless of the number of successful tool calls, so repeated execution does not increase the reward. An incorrect trajectory can receive at most 0.5 from the format term and receives no reward for code execution. This design encourages the policy to use executable analysis when it contributes to a correct answer, rather than rewarding program execution by itself.

Table 1: Results on the in-distribution test split of OmniMedVQA. We report accuracy in percent. Avg. is the unweighted mean over the four question categories. ✗ denotes no tool access, Fixed tools denotes the native predefined interfaces of prior agents, and General code denotes model-generated programs executed in MedCodeLab. Boldface marks the best result among non-proprietary models.

Method	Tool interface	Anatomy	Diagnosis	Grading	Other	Avg.
Proprietary VLMs						
GPT 5.2	✗	54.4	71.1	76.2	81.7	70.8
Gemini 3 Pro	✗	70.6	88.1	82.4	90.6	82.9
Gemini 3 Flash	✗	59.5	84.7	80.5	83.1	77.0
Gemini 3.5 Flash	✗	71.0	87.9	85.2	91.7	84.0
Claude Sonnet 4.5	✗	47.3	75.0	72.9	66.3	65.4
Open general and medical VLMs						
Qwen2.5-VL-7B	✗	40.4	57.5	55.2	55.1	52.1
InternVL3-8B	✗	63.0	83.3	72.4	79.4	74.5
Chiron-o1-8B	✗	59.4	80.5	75.7	74.6	72.5
MedVLThinker-7B	✗	44.2	62.7	68.1	55.1	57.5
Lingshu-I-8B	✗	66.6	78.7	70.5	80.6	74.1
Visual and medical agents						
DeepEyes-7B	Fixed tools	44.7	68.2	64.8	61.1	59.7
MMedAgent-7B	Fixed tools	46.3	64.2	56.2	51.1	54.5
Pixel-Reasoner-7B	Fixed tools	46.8	68.3	69.0	58.6	60.7
Backbone and our model						
Qwen3-VL-8B	✗	66.8	82.8	70.5	80.9	75.2
MedCode-8B	General code	**72.4**	**87.0**	**81.4**	**90.3**	**82.8**
Δ		**+5.6**	**+4.2**	**+10.9**	**+9.4**	**+7.6**

Policy Optimization. The trajectory rewards defined above determine how the sampled responses contribute to the policy update (Shao et al., 2024). For the i-th trajectory τ_i, we write its reward as $R_i = R(\tau_i)$. We normalize R_i using the mean $\overline{R}$ and population standard deviation s_R of the G rewards obtained for the same question. This gives the group-relative advantage $A_i = (R_i - \overline{R})/(s_R + 10^{-6})$, which is shared by all assistant tokens in τ_i. Let $\mathcal{A}(\tau_i)$ denote the positions of assistant-generated tokens in τ_i. For the token at position j, $\rho_{i,j}(\theta)$ denotes the likelihood ratio between the current policy π_θ and the rollout policy π_{old}. We keep the SFT model frozen as the reference policy π_{ref}. The term $d_{i,j}$ denotes the nonnegative samplewise KL penalty between π_θ and π_{ref} at the same token position. We optimize the policy with

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{A}(\tau_i)|}\sum_{j\in\mathcal{A}(\tau_i)}\Big(\min\big(\rho_{i,j}(\theta)A_i,\ \text{clip}(\rho_{i,j}(\theta),1-\epsilon,1+\epsilon)A_i\big)-\beta d_{i,j}\Big)\right], \tag{7}$$

where ϵ controls the clipping range and β controls the strength of the reference-policy regularization.

3 Experiments

3.1 Experimental Setup

Benchmarks and Metrics. We use the two test partitions defined in Section 2.3. The in-distribution split contains 7,747 questions and is evaluated by accuracy on anatomy, diagnosis, lesion grading, and other biological attributes. The source-held-out split contains 2,121 questions from BreakHis, PAD-UFES-20, PALM2019, and Chest X-Ray PA, all of which are excluded from training. We report accuracy for each source and their macro average. A prediction is correct only when a single final option can be extracted and matches the reference answer. Invalid or missing answers are counted as incorrect.

Implementation Details. We train MedCode on 16 NVIDIA H800 GPUs and initialize it from Qwen3-VL-8B-Instruct (Bai et al., 2025a). Supervised fine-tuning runs for 10 epochs over the

Table 2: Results on the source-held-out split of OmniMedVQA. We report accuracy in percent. Avg. is the unweighted mean over the four source accuracies. Boldface marks the best result among non-proprietary models.

Method	Tool interface	BreakHis	PAD-UFES-20	PALM2019	Chest X-Ray PA	Avg.
Proprietary VLMs						
GPT 5.2	✗	75.6	64.1	86.0	92.1	79.5
Gemini 3 Pro	✗	81.9	75.5	90.6	95.7	85.9
Gemini 3 Flash	✗	81.2	69.8	90.1	95.3	84.1
Gemini 3.5 Flash	✗	83.4	77.5	91.9	97.0	87.4
Claude Sonnet 4.5	✗	73.3	58.9	82.2	88.3	75.7
Open general and medical VLMs						
Qwen2.5-VL-7B	✗	60.8	47.3	64.6	71.0	60.9
InternVL3-8B	✗	77.1	65.4	79.6	86.3	77.1
Chiron-o1-8B	✗	72.4	61.8	77.1	83.1	73.6
MedVLThinker-7B	✗	62.1	49.6	70.5	71.3	63.4
Lingshu-I-8B	✗	78.5	70.5	80.2	87.4	79.2
Visual and medical agents						
DeepEyes-7B	Fixed tools	65.0	56.1	78.6	83.6	70.8
MMedAgent-7B	Fixed tools	55.8	49.6	63.4	76.9	61.4
Pixel-Reasoner-7B	Fixed tools	65.3	51.9	76.3	83.2	69.2
Backbone and our model						
Qwen3-VL-8B	✗	74.4	67.2	76.6	86.8	76.2
MedCode-8B	General code	**81.0**	**76.5**	**90.1**	**94.4**	**85.5**
Δ		**+6.6**	**+9.3**	**+13.5**	**+7.6**	**+9.3**

49,817 trajectories in MedCode-50k, with a peak learning rate of 1×10^{-5} and an effective global batch size of 256 trajectories. We then run reinforcement learning for 5 epochs over 15,895 selected questions with a peak learning rate of 1×10^{-6}. Each RL rollout batch contains 32 questions. GRPO samples eight trajectories for each question, giving 256 candidate trajectories per batch. The learning rate follows cosine decay after a 3% linear warm-up. We use a 32,768-token context window and allow at most 8,192 generated tokens, and 60 seconds for each program execution. Evaluation uses greedy decoding.

3.2 Main Results

We compare MedCode with proprietary VLMs (OpenAI, 2025; Google DeepMind, 2025b;a; 2026; Anthropic, 2025), open general and medical VLMs (Bai et al., 2025b; Zhu et al., 2025; Sun et al., 2026; Huang et al., 2025; Bai et al., 2025a; Xu et al., 2026), and tool-augmented agents (Zheng et al., 2026; Li et al., 2024; Su et al., 2026).

In-distribution Results. As shown in Table 1, MedCode achieves an average accuracy of 82.8%, yielding a 7.6 absolute gain over its Qwen3-VL-8B baseline. Improvements appear across all four question categories. The largest gain is observed for lesion grading at 10.9, followed by other biological attributes at 9.4, anatomy at 5.6, and diagnosis at 4.2. MedCode also outperforms the strongest open model without tools, InternVL3-8B, by 8.3 in average accuracy. Despite using only 8B parameters, MedCode remains close to Gemini 3 Pro and Gemini 3.5 Flash, whose average accuracies are 82.9% and 84.0%. The consistent improvement across categories indicates that general code execution benefits a broad range of medical visual reasoning tasks.

Generalization to Unseen Sources. Table 2 further examines transfer to datasets excluded from both stages of training. MedCode obtains an average of 85.5%, improving its baseline by 9.3. The gains are consistent across all four sources and are particularly pronounced on PALM2019 PAD-UFES-20, and Chest X-Ray PA, where accuracy improves by 13.5, 9.3, and 7.6, respectively. MedCode slightly surpasses Gemini 3 Flash. Overall, the source-held-out results suggest that the learned interaction with MedCodeLab transfers beyond the datasets used for trajectory collection.

Table 3: Effects of program scope, execution reward, and feedback channel. Accuracy is reported on the in-distribution and source-held-out test sets.

(a) Program scope

Setting	ID	Held-out
Crop and resize	77.5	80.0
Fixed operations	79.9	81.8
General code	**82.8**	**85.5**

(b) Execution reward

Setting	ID	Held-out
No execution term	81.4	84.1
Ungated execution	80.7	83.9
Correctness-gated	**82.8**	**85.5**

(c) Feedback channel

Setting	ID	Held-out
Text only	80.5	83.5
Image only	81.3	84.0
Text and image	**82.8**	**85.5**

3.3 Ablation Studies

We organize the ablations around the two choices that distinguish MedCode from existing tool-using systems. The first is to express image analysis as general-purpose code rather than a prescribed set of operations. The second is to learn this behavior from complete execution trajectories and then refine it through interaction with MedCodeLab. We further examine the reward and feedback used during this process. Unless noted otherwise, all variants use the same evaluation examples and inference budget as the main experiments.

Beyond Predefined Image Operations. We first restrict the operations available to the final MedCode checkpoint progressively. The most constrained variant can only crop and resize the input. The next variant adds a fixed collection of commonly used operations, including rotation, contrast adjustment, thresholding, histograms, edge detection, and connected-component analysis. The full model retains unrestricted access to MedCodeLab. These restrictions are stated in the prompt and enforced by the environment, while the interaction budget remains unchanged. Table 3 (a) shows a clear progression as the available analysis becomes more flexible. Crop and resize alone reach 77.5% on the in-distribution split and 80.0% on the held-out split. A fixed collection of image operations closes much of the gap, but still falls short of general code on both splits. Iterative access to the image therefore accounts for a substantial part of the improvement, while the remaining gap supports the motivation introduced in Section 1. The ability to define a computation for the question at hand offers additional value beyond composing a predefined list of operations.

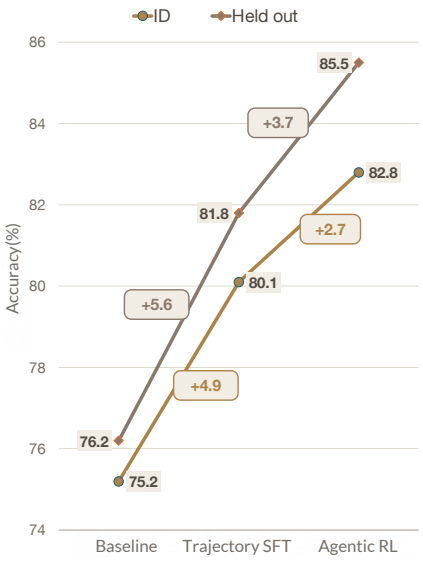


Figure 2: Accuracy after supervised fine-tuning and reinforcement learning.

Learning from Execution Trajectories. We next trace how the ability to use MedCodeLab develops over the two training stages. Figure 2 begins with the Qwen3-VL-8B baseline evaluated without tool access. Supervised fine-tuning on MedCode-50k raises in-distribution accuracy from 75.2% to 80.1%, while held-out accuracy rises from 76.2% to 81.8%. Reinforcement learning further improves the two results to 82.8% and 85.5%. Most of the initial improvement comes from learning the execution trajectories in MedCode-50k. Online interaction then provides a second, sizeable gain by allowing the policy to explore its own programs and learn from their outcomes. The similar trend on held-out sources suggests that this training does more than fit the datasets used to collect the trajectories.

Rewarding Useful Execution. Since reinforcement learning provides a meaningful improvement, we examine whether the execution term in Eq. 6 contributes to that result. All variants start from the same SFT checkpoint and differ only in how successful execution is rewarded. Results are in Table 3 (b). Removing the execution term produces results close to the full objective. In contrast, rewarding every successful program regardless of the final answer lowers accuracy to 80.7% in distribution and 83.9% on held-out sources. The correctness-gated reward recovers the best result on both splits. Its main role is therefore not to reward tool use more often, but to avoid reinforcing programs that run successfully without helping the model reach the correct answer. This behavior is consistent with the reward design in Section 2.4, where execution is treated as useful only when it supports the final prediction.

Textual and Visual Execution Feedback. Finally, we vary the observation returned by MedCodeLab. The text-only setting removes derived images, and the image-only setting removes computed textual outputs. Both retain the execution status, error messages when needed, and the original input image. As shown in Table 3 (c), the complete feedback gives the best accuracy. Image-only feedback remains within 1.5 percentage points of the full setting on both splits, while text-only feedback gives 80.5% and 83.5%. Derived images appear slightly more informative for these benchmarks, but neither result points to a dependence on one output format. MedCode can use measurements returned as text as well as visual evidence produced during execution, which is the multimodal interaction that MedCodeLab was designed to support.

4 Related Works

4.1 Medical Vision-Language Reasoning

Medical VLMs support image-based question answering, diagnosis, and report generation (Moor et al., 2023; Luo et al., 2023; Li et al., 2023a; Pellegrini et al., 2023; Chen et al., 2024b; Tu et al., 2024). Recent work further trains these models to produce explicit reasoning. MedVLM-R1 and Med-R1 use reinforcement learning to improve medical reasoning and generalization (Pan et al., 2025; Lai et al., 2026). Chiron-o1 constructs reasoning paths through collaborative search and curriculum learning (Sun et al., 2026), while MedVLThinker compares distilled rationales with reinforcement learning using verifiable rewards (Huang et al., 2025). Together, these approaches move medical VLMs beyond direct prediction and show that structured supervision can improve both answer accuracy and the coherence of intermediate decisions. Despite these advances, reasoning remains primarily textual. The image is generally encoded once, after which the model uses a fixed visual representation (Zhang et al., 2023). It therefore cannot acquire new measurements or transformed image evidence. This limitation is important when an answer depends on lesion extent, intensity variation, boundary morphology, or comparison between several regions. MedCode instead places executable image analysis inside the reasoning loop and continues from results obtained directly from the image.

4.2 Tool-Augmented Visual Reasoning

Tool-augmented models allow external operations to modify or re-examine images during inference (Wu et al., 2023; Surís et al., 2023; Gupta & Kembhavi, 2023). DeepEyes learns active region inspection through reinforcement learning (Zheng et al., 2026). VTool-R1 interleaves reasoning with Python-based visual editing tools (Wu et al., 2026). Medical agents have followed a related direction. MMedAgent and AURA combine VLMs with specialist models (Li et al., 2024; Fathi et al., 2025). Med-VRAgent uses visual guidance and tree search (Guo et al., 2025), while Ophiuchus and MedVistaGym train models to invoke predefined tools and reason from returned observations (Jiang et al.; Lu et al., 2026). These systems acquire new visual evidence, but their analyses remain bounded by tools defined in advance. Such procedures are difficult to represent as a sequence of fixed, isolated tool calls. MedCode instead generates question-specific programs from general image-processing and numerical operations. A shared environment can therefore support region inspection, measurement, comparison, and multi-step computation across medical modalities without specialist perception models.

5 Conclusion

We introduced MedCode, a framework that enables medical vision language models to construct question specific image analyses through executable code. MedCodeLab provides a shared environment for executing these programs and returning textual and visual evidence for subsequent reasoning. We also constructed MedCode-50k with 49,817 verified trajectories across eight imaging modalities and four question categories. MedCode is trained through supervised fine tuning on these trajectories, followed by reinforcement learning with the execution environment. MedCode achieves 82.8% accuracy on the in distribution test set and 85.5% across four unseen data sources. Ablation studies show that execution trajectory training provides substantial improvements, while reinforcement learning further strengthens the model. General code also performs better than crop based reasoning and predefined image operations. These findings suggest that executable code offers a flexible and effective approach to medical visual reasoning.

AI USE STATEMENT

In this work, we used Gemini 3.1 Pro to generate synthetic multi-turn training trajectories for MedCode-50k, including reasoning, executable programs, and final answers. Execution feedback and derived images were obtained by running the generated programs in MedCodeLab. Reference answers were withheld from the teacher during generation and used for subsequent answer verification. We retained 49,817 training trajectories after applying the answer-verification, execution, interaction-format, and quality filters described in Section 2.3 and Appendix B.2.

We additionally used generative AI tools to assist with language editing and refinement of the manuscript, including improvements to grammar, clarity, and readability. All revisions made with AI assistance were reviewed by the authors, and the technical content was checked against the implemented methods and reported experiments. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Anthropic. Claude Sonnet 4.5 System Card. System card, Anthropic, 2025. URL https://www.anthropic.com/claude-sonnet-4-5-system-card.

Amanullah Asraf and Zabirul Islam. Covid19, pneumonia and normal chest x-ray pa dataset. *Mendeley Data*, 1(2), 2021.

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Michael Baumgartner, Paul F Jäger, Fabian Isensee, and Klaus H Maier-Hein. nndetection: a self-configuring method for medical object detection. In *International conference on medical image computing and computer-assisted intervention*, pp. 530–539. Springer, 2021.

Junying Chen, Chi Gui, Ruyi Ouyang, Anningzhe Gao, Shunian Chen, Guiming Hardy Chen, Xidong Wang, Zhenyang Cai, Ke Ji, Xiang Wan, et al. Towards injecting medical visual knowledge into multimodal llms at scale. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7346–7370, 2024a.

Zhihong Chen, Maya Varma, Jean-Benoit Delbrouck, Magdalini Paschali, Louis Blankemeier, Dave Van Veen, Jeya Maria Jose Valanarasu, Alaa Youssef, Joseph Paul Cohen, Eduardo Pontes Reis, et al. Chexagent: Towards a foundation model for chest x-ray interpretation. In *AAAI 2024 Spring Symposium on Clinical Foundation Models*, 2024b.

Huihui Fang, Fei Li, Junde Wu, Huazhu Fu, Xu Sun, José Ignacio Orlando, Hrvoje Bogunović, Xiulan Zhang, and Yanwu Xu. Open fundus photograph dataset with pathologic myopia recognition and anatomical structure annotation. *Scientific data*, 11(1):99, 2024.

Nima Fathi, Amar Kumar, and Tal Arbel. Aura: A multi-modal medical agent for understanding, reasoning and annotation. In *International Workshop on Agentic AI for Medicine*, pp. 105–114. Springer, 2025.

David T Felson. Osteoarthritis of the knee. *New England journal of medicine*, 354(8):841–848, 2006.

Google DeepMind. Gemini 3 Flash Model Card. Model card, Google DeepMind, 2025a. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf.

Google DeepMind. Gemini 3 Pro Model Card. Model card, Google DeepMind, 2025b. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf. Updated May 2026.

Google DeepMind. Gemini 3.5 Flash Model Card. Model card, Google DeepMind, 2026. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-5-Flash-Model-Card.pdf.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14375–14385. IEEE, 2024.

Guangfu Guo, Xiaoqian Lu, and Yue Feng. Med-vragent: a framework for medical visual reasoning-enhanced agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18613–18627, 2025.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962. IEEE, 2023.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the association for computational linguistics: ACL 2023*, pp. 8003–8017, 2023.

Yutao Hu, Tianbin Li, Quanfeng Lu, Wenqi Shao, Junjun He, Yu Qiao, and Ping Luo. Omnimedvqa: A new large-scale comprehensive evaluation benchmark for medical lvlm. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22170–22183. IEEE, 2024.

Xiaoke Huang, Juncheng Wu, Hui Liu, Xianfeng Tang, and Yuyin Zhou. Medvlthinker: Simple baselines for multimodal medical reasoning. *arXiv preprint arXiv:2508.02669*, 2025.

Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

Yankai Jiang, Yujie Zhang, Peng Zhang, Wenjie Li, Yichen Li, Jintai Chen, Xiaoming Shi, and Shihui Zhen. Ophiuchus: Incentivizing tool-augmented "think with images" for joint medical segmentation, understanding and reasoning.

Yuxiang Lai, Jike Zhong, Ming Li, Shitian Zhao, Yuheng Li, Konstantinos Psounis, and Xiaofeng Yang. Med-r1: Reinforcement learning for generalizable medical reasoning in vision-language models. *IEEE transactions on medical imaging*, 2026.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. Mitigating object hallucinations in large vision-language models through visual contrastive decoding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13872–13882. IEEE, 2024.

Binxu Li, Tiankai Yan, Yuanting Pan, Jie Luo, Ruiyang Ji, Jiayuan Ding, Zhe Xu, Shilong Liu, Haoyu Dong, Zihao Lin, et al. Mmedagent: Learning to use medical tools with multi-modal agent. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8745–8760, 2024.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Advances in neural information processing systems*, 36:28541–28564, 2023a.

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

Ruhan Liu, Xiangning Wang, Qiang Wu, Ling Dai, Xi Fang, Tao Yan, Jaemin Son, Shiqi Tang, Jiang Li, Zijian Gao, et al. Deepdrid: Diabetic retinopathy—grading and image quality estimation challenge. *Patterns*, 3(6), 2022.

Meng Lu, Yuxing Lu, Yuchen Zhuang, Megan Mullins, Yang Xie, Guanghua Xiao, Charles Fleming, Wenqi Shi, and Xuan Wang. Medvistagym: A scalable training environment for thinking with medical images via tool-integrated reinforcement learning. *arXiv preprint arXiv:2601.07107*, 2026.

Yizhen Luo, Jiahuan Zhang, Siqi Fan, Kai Yang, Yushuai Wu, Mu Qiao, and Zaiqing Nie. Biomedgpt: Open multimodal generative pre-trained transformer for biomedicine. *arXiv preprint arXiv:2308.09442*, 2023.

Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature communications*, 15(1):654, 2024.

Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Yash Dalmia, Jure Leskovec, Cyril Zakka, Eduardo Pontes Reis, and Pranav Rajpurkar. Med-flamingo: a multimodal medical few-shot learner. In *Machine learning for health (ML4H)*, pp. 353–367. PMLR, 2023.

OpenAI. Update to GPT-5 System Card: GPT-5.2. System card, OpenAI, 2025. URL https://cdn.openai.com/pdf/3a4153c8-c748-4b71-8e31-aecbde944f8d/oai_5_2_system-card.pdf.

Andre GC Pacheco, Gustavo R Lima, Amanda S Salomao, Breno Krohling, Igor P Biral, Gabriel G De Angelo, Fábio CR Alves Jr, José GM Esgario, Alana C Simora, Pedro BC Castro, et al. Pad-ufes-20: A skin lesion dataset composed of patient data and clinical images collected from smartphones. *Data in brief*, 32:106221, 2020.

Jiazhen Pan, Che Liu, Junde Wu, Fenglin Liu, Jiayuan Zhu, Hongwei Bran Li, Chen Chen, Cheng Ouyang, and Daniel Rueckert. Medvlm-r1: Incentivizing medical reasoning capability of vision-language models (vlms) via reinforcement learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 337–347. Springer, 2025.

Chantal Pellegrini, Ege Özsoy, Benjamin Busam, Nassir Navab, and Matthias Keicher. Radialog: A large vision-language model for radiology report generation and conversational assistance. *arXiv preprint arXiv:2311.18681*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

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.

Fabio A Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. A dataset for breast cancer histopathological image classification. *Ieee transactions on biomedical engineering*, 63 (7):1455–1462, 2015.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. *Advances in Neural Information Processing Systems*, 38:8222–8251, 2026.

Haoran Sun, Yankai Jiang, Wenjie Lou, Yujie Zhang, Wenjie Li, Lilong Wang, Mianxin Liu, Lei Liu, and Xiaosong Wang. Chiron-o1: Igniting multimodal large language models towards generalizable medical reasoning via mentor-intern collaborative search. *Advances in Neural Information Processing Systems*, 38:104180–104217, 2026.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11854–11864. IEEE, 2023.

Tao Tu, Shekoofeh Azizi, Danny Driess, Mike Schaekermann, Mohamed Amin, Pi-Chuan Chang, Andrew Carroll, Charles Lau, Ryutaro Tanno, Ira Ktena, et al. Towards generalist biomedical ai. *Nejm Ai*, 1(3):AIoa2300138, 2024.

Joost JM Van Griethuysen, Andriy Fedorov, Chintan Parmar, Ahmed Hosny, Nicole Aucoin, Vivek Narayan, Regina GH Beets-Tan, Jean-Christophe Fillion-Robin, Steve Pieper, and Hugo JWL Aerts. Computational radiomics system to decode the radiographic phenotype. *Cancer research*, 77(21):e104, 2017.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. *arXiv preprint arXiv:2402.01030*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. arxiv 2023. *arXiv preprint arXiv:2303.04671*, 10, 2023.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. In *International Conference on Learning Representations*, volume 2026, pp. 78298–78319, 2026.

Weiwen Xu, Hou Pong Chan, Long Li, Mahani Aljunied, Ruifeng Yuan, Jianyu Wang, Chenghao Xiao, Guizhen Chen, Chaoqun Liu, Zhaodonghui Li, et al. Lingshu: Generalist foundation model for unified multimodal medical understanding and reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

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. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, and Chao Shen. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 126775–126798, 2026.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Table 4: Category-wise ablations on the ID test split. Accuracy is expressed in percent.

Configuration	Anatomy	Diagnosis	Grading	Other	Avg.
(a) Program scope					
Crop and resize	70.2	81.7	75.7	82.6	77.5
Fixed operations	71.0	83.6	77.1	88.0	79.9
General code	72.4	87.0	81.4	90.3	82.8
(b) Training stages					
Qwen3-VL-8B (no tools)	66.8	82.8	70.5	80.9	75.2
MedCode-SFT	72.0	85.5	76.7	86.3	80.1
MedCode (SFT + RL)	72.4	87.0	81.4	90.3	82.8
(c) Execution reward					
No execution term	72.2	86.3	79.0	88.3	81.4
Ungated execution	72.1	85.6	77.6	87.4	80.7
Correctness-gated	72.4	87.0	81.4	90.3	82.8
(d) Feedback channel					
Text only	71.8	85.8	75.7	88.6	80.5
Image only	72.2	86.6	78.6	87.7	81.3
Text and image	72.4	87.0	81.4	90.3	82.8

Table 5: Source-wise ablations on the source-held-out test split.

Configuration	BreakHis	PAD-UFES-20	PALM2019	Chest X-Ray PA	Avg.
(a) Program scope					
Crop and resize	76.0	69.8	83.7	90.5	80.0
Fixed operations	77.6	71.3	85.5	92.6	81.8
General code	81.0	76.5	90.1	94.4	85.5
(b) Training stages					
Qwen3-VL-8B (no tools)	74.4	67.2	76.6	86.8	76.2
MedCode-SFT	78.0	70.0	87.5	91.7	81.8
MedCode (SFT + RL)	81.0	76.5	90.1	94.4	85.5
(c) Execution reward					
No execution term	79.4	73.9	89.1	94.2	84.1
Ungated execution	79.1	74.4	88.3	93.8	83.9
Correctness-gated	81.0	76.5	90.1	94.4	85.5
(d) Feedback channel					
Text only	78.3	74.7	88.0	92.9	83.5
Image only	79.2	75.2	88.8	92.8	84.0
Text and image	81.0	76.5	90.1	94.4	85.5

A Additional Quantitative Analyses

Tables 4 and 5 expand the ablations in Section 3.3 to their individual evaluation components. The first reports Anatomy, Diagnosis, Grading, and Other on the 7,747-question in-distribution (ID) split. The second reports BreakHis, PAD-UFES-20, PALM2019, and Chest X-Ray PA on the 2,121-question source-held-out split. Each table includes all conditions for program scope, training stage, execution reward, and feedback channel. Avg. is the unweighted mean of its four component columns.

B Details of MedCode-50k

B.1 Source Coverage and Partition Accounting

MedCode-50k contains the accepted training trajectories, rather than all questions in the source pool. The local open-access OmniMedVQA annotations contain 88,996 question–answer pairs from 42

sources. Excluding all 11,565 Modality Recognition questions leaves 77,431 pairs in four question categories and 40 sources. DeepDRiD (Liu et al., 2022) and Knee Osteoarthritis (Felson, 2006) contribute only modality-recognition questions in these annotations, so neither remains after task selection. This is a task-filtered subset of the open-access data, not the full OmniMedVQA benchmark.

B.2 Trajectory Generation and Filtering

Gemini 3.1 Pro receives the image, question, and the interaction prompt in Appendix C.2. The reference answer is kept outside the model-visible conversation. Rejection sampling first discards trajectories whose final answer is missing, ambiguous, or inconsistent with the reference. The remaining trajectories are screened for four quality failures: (a) no successful code execution; (b) exhaustion of the interaction budget without a valid terminal response; (c) trivial programs that do not analyze image data or computed quantities; and (d) malformed interaction structure. A failed intermediate call is allowed if a later call recovers successfully. Conversely, a correct final option does not rescue an otherwise invalid trajectory.

B.3 MedCode-50k Trajectory Examples

A MedCode-50k example is a complete interaction trajectory, not an image–question–answer triple. The examples below therefore show the initial question and image, assistant rationale and program, textual execution outputs, the returned images, and the terminal answer in their recorded order. User and environment messages provide context, while assistant messages supply the SFT targets described in Eq. 4. We present each example as a compact question–analysis–feedback–answer sequence. Code excerpts preserve the original computational statements, while connecting prose summarizes intermediate inspection, retries, and extended rationales. Routine setup is omitted from the excerpts. The shared interaction prompt appears in Listing 1.

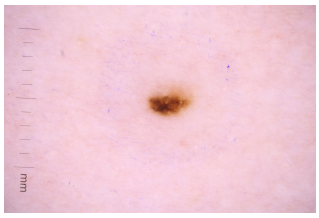
B.3.1 Example 1: Lesion Symmetry and Ruler-Based Measurement

Source: ISIC2020, question 0491. **Modality:** Dermoscopy. **Recorded tool calls:** 8. We highlight how the teacher develops complementary measurements and refines its scale estimate, retaining the key computations from calls 1 and 7.

Question and input

Q: Is the abnormality in this image considered as a benign or malignant?

Options: A. Benign image. B. Malignant.



Original image



Returned lesion crop (call 3)



Returned ruler crop (call 3)

Key code and observations

Call 1: turn a visual criterion into a computable quantity. The teacher operationalizes lesion symmetry as overlap between a centered lesion mask and its horizontal and vertical reflections. This produces explicit, inspectable measurements of two symmetry axes, going beyond a verbal judgment from an enlarged crop. The excerpt retains the computation after mask construction and validity checks:

```
flipped_lr = np.fliplr(cropped_mask)
flipped_ud = np.flipud(cropped_mask)
intersection_lr = np.logical_and(cropped_mask, flipped_lr).sum()
union_lr = np.logical_or(cropped_mask, flipped_lr).sum()
iou_lr = intersection_lr / union_lr if union_lr > 0 else 0
intersection_ud = np.logical_and(cropped_mask, flipped_ud).sum()
```

```
union_ud = np.logical_or(cropped_mask, flipped_ud).sum()
iou_ud = intersection_ud / union_ud if union_ud > 0 else 0
```

Returned values: left–right IoU $= 0.8348$; up–down IoU $= 0.7971$; circularity $= 0.5737$.

Calls 2–6: use feedback to adapt the analysis. The teacher checks the segmentation and obtains lesion/ruler crops. It subsequently obtains an initial 0.59 mm estimate, then continues refining the ruler analysis. This sequence shows that execution feedback can guide changes to the analytical procedure, rather than merely confirm a fixed sequence of tool calls.

Call 7: derive the measurement scale from the image. The teacher narrows the ruler strip, extracts peaks from its intensity projection, and uses their median spacing to convert the previously measured 781×434-pixel lesion bounding box into a physical-size estimate. The same program links reference-object detection to lesion measurement:

```
ruler_slice = img[:, 50:150]
ruler_inv = 255 - ruler_slice
projection = np.mean(ruler_inv, axis=1)
peaks, properties = find_peaks(projection, prominence=5, distance=100)
if len(peaks) > 1:
    distances = np.diff(peaks)
    median_dist = np.median(distances)
    width_mm = 781 / median_dist
    height_mm = 434 / median_dist
```

Returned values: 9 detected peaks; median spacing 183.5 pixels; estimated dimensions 4.26×2.37 mm, under the program's assumption of 1 mm between adjacent ruler ticks.

Call 8: combine complementary evidence. Mask-conditioned color matching returns two categories, `Light Brown` and `Dark Brown`. A further spatial edge-density analysis returns an overall density of 0.0013 and compares its distribution across lesion quadrants. The teacher therefore combines shape, size, color, and spatial structure in its final interpretation; this last program and the extended rationale are condensed here.

Outcome and methodological value

The teacher combines these observations and returns `<answer>A</answer>`, matching the reference option. The value of the trajectory lies in its construction of a question-specific analysis: the model defines the symmetry statistic, extracts a measurement scale from the image, and supplements these quantities with color and structure comparisons. A larger menu of fixed operations can provide individual transformations, but here the model itself specifies how their outputs become measurements relevant to the question. The recorded revisions further demonstrate the adaptive interaction that MedCode-50k is designed to teach.

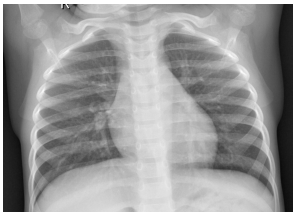
B.3.2 Example 2: Boundary Search and Cardiothoracic Measurement

Source: OCT & X-Ray 2017, question 0987. **Modality:** Chest radiography. **Recorded tool calls:** 3. We highlight the progression from an image profile to a custom boundary-search and measurement loop.

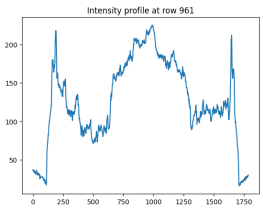
Question and input

Q: Is there any anomaly detected in this particular image?

Options: A. No, this image suggests the presence of emphysema. B. Yes, there is a large mass visible in the lungs on this image. C. No, this image displays an enlarged heart. D. No, this is a normal chest X-ray image.



Original image



Returned intensity profile (call 2)

Key code and observations

Calls 1–2: expose the image structure for measurement. The teacher inspects the 1282×1788-pixel image and returns an intensity profile at 75% of its height. This makes the intensity transitions available as a derived observation before it constructs the width-measurement procedure. The inspection and plotting code are condensed here.

Call 3: construct the measurement algorithm. Rather than requesting a predefined cardiothoracic-ratio tool, the teacher writes a scan over rows from 50% to 85% of the image height, at 10-pixel intervals. The program smooths each profile, defines separate gradient-search intervals for candidate thoracic and cardiac boundaries, and retains the maximum widths across rows. Thus, the search strategy and aggregation rule are specified within the generated program, not supplied as a packaged measurement routine. Image loading, initialization, and final print statements are omitted below.

```
for y in range(start_y, end_y, 10):
    row = img[y, :]
    smoothed_row = np.convolve(row, np.ones(20)/20, mode='same')
    gradient = np.gradient(smoothed_row)
    left_thoracic_edge = np.argmax(gradient[:int(width*0.3)])
    right_thoracic_edge = int(width*0.7) + np.argmin(gradient[int(width*0.7):])
    thoracic_width = right_thoracic_edge - left_thoracic_edge
    if thoracic_width > max_thoracic_width:
        max_thoracic_width = thoracic_width
    left_heart_search = gradient[int(width*0.3):int(width*0.5)]
    right_heart_search = gradient[int(width*0.5):int(width*0.7)]
    if len(left_heart_search) > 0 and len(right_heart_search) > 0:
        left_heart_edge = int(width*0.3) + np.argmax(left_heart_search)
        right_heart_edge = int(width*0.5) + np.argmin(right_heart_search)
        heart_width = right_heart_edge - left_heart_edge
        if heart_width > max_heart_width:
            max_heart_width = heart_width
```

Returned values: maximum thoracic width $= 1578$ pixels; maximum heart width $= 675$ pixels; estimated ratio $675/1578 = 0.428$ (rounded).

Outcome and methodological value

The teacher uses the computed ratio alongside its inspection of the original image and returns `<answer>D</answer>`, matching the reference. This case illustrates the transition from obtaining another view to constructing a task-specific estimator: boundary selection, iteration across rows, maximum-width aggregation, and ratio computation are all expressed in code. The resulting calculation is inspectable and can be revised at the level of its search regions or aggregation logic, without changing MedCodeLab or introducing a new specialist tool.

Together, these examples illustrate the two motivations in Section 1. Analytical flexibility comes from specifying task-dependent statistics, search regions, and aggregation rules directly in code. Extensibility comes from expressing both dermoscopic assessment and radiographic measurement through the same MedCodeLab interface, without adding a dedicated lesion-assessment or cardiothoracic-ratio tool. These trajectories also expose a useful training signal: the model learns not only to produce an answer, but to construct an analysis, inspect its outputs, and revise the procedure when additional evidence is needed.

C MedCodeLab and Interaction Prompts

C.1 Execution Interface and Observation Construction

MedCodeLab implements the interaction in Eq. 1 through a single `code_interpreter` entry point. The collector provides standard mathematical and image processing utilities. This is a library-level interface: the policy writes the analytical program, rather than passing arguments to a specialist diagnostic model. Programs operate on the available source image file.

Inputs, State, and File Handling. The initial user message supplies an anonymized image filename together with the image and question. The execution runner exposes `IMAGE_PATH` and `OUTPUT_DIR`, and starts each call in a fresh temporary working directory and Python process. Python variables therefore do not persist between calls. The policy must reconstruct required intermediate computations rather than assume an interactive notebook state. Conversation history,

including previously returned images and text, persists at the policy level. The archived collector uses a UUID-based filename instead of presenting the original label-bearing directory name in the user prompt.

Execution Feedback. The runner captures standard output, diagnostic output, and generated image files. It also collects figures displayed through supported plotting and image-display functions. The resulting text and image items are enclosed by `<tool_response>` delimiters and appended before the next assistant turn. In the archived conversation format these observations are stored as user-role multimodal messages, but originate from the environment rather than the human question. A runtime exception or timeout is returned as an error observation; it is not silently converted into an empty successful result. Assistant text appearing after the tool-call boundary is not treated as execution evidence.

Budgets and Execution Boundaries. The experimental setup specifies a 60-second execution timeout, a 32,768-token context window, and an 8,192-token generation budget. The available collector entry point additionally defaults to 10 assistant turns, counting a terminal answer as a turn.

C.2 System and User Prompts

Listing 1 reproduces the system prompt shipped with the available trajectory collector. Its tool-call grammar matches the executable interface in Section 2.

Listing 1: System prompt from the archived trajectory collector, reproduced verbatim.

```
# Tools

You have access to the following functions:

<tools>
{"type": "function", "function": {"name": "code_interpreter", "description": "Python code
     sandbox, which can be used to execute Python code.", "parameters": {"type": "object", "
     properties": {"code": {"description": "The python code.", "type": "string"}}, "required":
      ["code"]}}}
</tools>

If you choose to call a function ONLY reply in the following format with NO suffix:

<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter
that can span
multiple lines
</parameter>
</function>
</tool_call>

<IMPORTANT>
Reminder:
- Function calls MUST follow the specified format: an inner <function=...></function> block
     must be nested within <tool_call></tool_call> XML tags
- Required parameters MUST be specified
- You may provide optional reasoning for your function call in natural language BEFORE the
     function call, but NOT after
- If there is no function call available, answer the question like normal with your current
     knowledge and do not tell the user about function calls
</IMPORTANT>

You are an expert medical image analyst. You MUST use the code interpreter to verify all
     answers | never rely on visual inspection alone.

# Methodology

**Observe** | Examine the image carefully.

**Verify programmatically (REQUIRED)** | Use `code_interpreter` to help solve the problem
     before drawing conclusions. Visual perception is unreliable for fine details, e.g.,
     spatial relationships, and counting. Before writing code, briefly state WHY you chose
     this particular image processing operation for this specific medical question.

**Answer** | Only after programmatic verification confirms your conclusion, state your final
     answer inside `&lt;answer&gt;&lt;/answer&gt;` tags, e.g., `&lt;answer&gt;B&lt;/answer&gt;`..
```

The user message consists of the original question, followed by the anonymized image filename. The corresponding image is attached as a separate multimodal item. The following is a serialization example:

```
What can be observed in this image?
A: Soft tissue swelling
B: Muscle strain
C: Tendonitis
D: Meniscal abnormality
Image file: <anonymized-image>.png
[The corresponding image is attached as an image item.]
```

The final option is emitted as a single `<answer>D</answer>` tag in this example. The reference option is not part of the user message.

C.3 Interfaces for the Ablations

The restricted conditions in Tables 4 and 5 are defined by the analytical operations available to the same trained checkpoint. The following specifies the intended interface contract.

Crop and Resize. We wrap image loading, rectangular cropping, and resizing as predefined tools that return the resulting images. The model can only invoke these tools, specifying crop coordinates and output sizes through their exposed arguments. It may inspect the returned images and make further calls, but cannot submit arbitrary code or perform operations outside this toolkit, such as intensity statistics, segmentation, or contrast enhancement. This condition preserves iterative local inspection while removing the ability to construct new image-analysis procedures.

Fixed Operations. We extend the crop-and-resize toolkit with individually wrapped tools for rotation, contrast adjustment, thresholding, intensity histograms, edge detection, and connected-component analysis. The model is restricted to calling tools from this predefined collection. It may select a tool, set its exposed parameters, and use returned outputs in subsequent calls, but cannot modify the underlying implementations, define additional tools, or execute custom analytical code. This condition provides a broader range of image-processing capabilities without specialist medical perception models, testing whether selecting and composing a richer fixed toolkit is sufficient compared with constructing question-specific computations in general code.

General Code. The full condition allows compositions expressed directly in the supported Python libraries, including custom intermediate arrays, measurements, and control flow. This flexibility is the intended distinction from a larger fixed tool menu. All three conditions retain the same image inputs, checkpoint, answer format, and overall inference budget.

Prompt Changes. For each restricted condition, the prompt replaces the general-code instructions with descriptions of the available tools and their arguments, while the question, options, and final-answer syntax remain unchanged. The crop condition states, "Use only the provided image-loading, cropping, and resizing tools; arbitrary code execution is unavailable." The fixed-operation condition states, "Use only the provided predefined image-processing tools; do not define new tools or execute custom code."

D Qualitative Results

D.1 Execution-Grounded Case Studies

We examine how a program changes the evidence available for the next reasoning step, rather than treating a plausible rationale as evidence of successful image analysis. Figure 3 uses saved artifacts from two teacher trajectories. The numerical outputs, derived images, and final options are taken from the recorded conversations. They illustrate the training-data format and the capabilities of the interface.

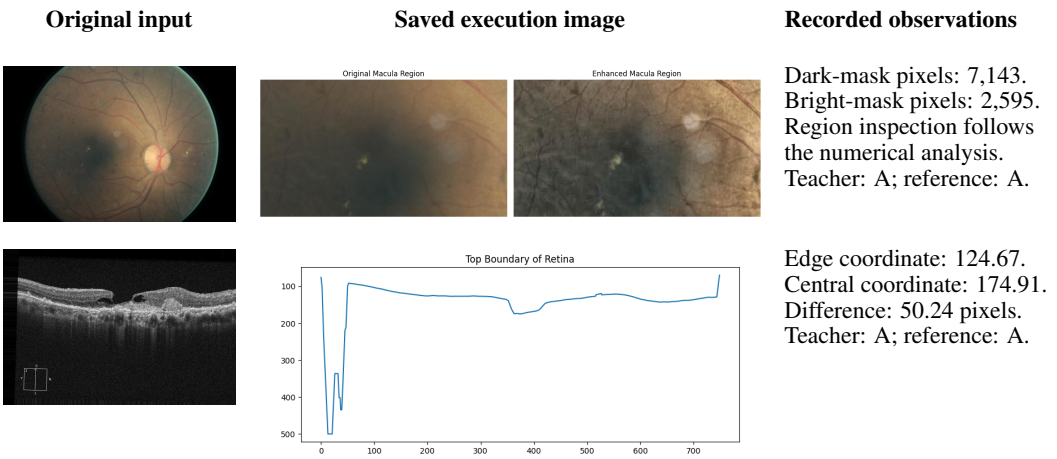


Figure 3: Trajectories combining numerical and visual feedback. Top: Diabetic Retinopathy question; bottom: Retinal OCT-C8 question.

Measurements Followed by Visual Inspection. The fundus example illustrates how generated code combines numerical analysis with targeted visual inspection. The teacher first constructs dark- and bright-pixel masks and counts the selected pixels, converting image appearance into explicit numerical evidence. It then extracts a region and generates an enhanced view, bringing local visual detail into the same reasoning sequence. MedCodeLab supports both steps through a shared execution interface, returning measurements and images for subsequent reasoning. The resulting trajectory extends crop-based inspection with model-defined computations, allowing quantitative summaries and visual observations to complement one another.

A Question-Specific Image Profile. The OCT example illustrates how general code turns a visual question into an explicit measurement procedure. The generated program combines thresholding, boundary-profile extraction, and a comparison of central and peripheral image coordinates, producing both a spatial profile and a numerical difference. Rather than invoking a prepackaged measurement tool, the teacher specifies the intermediate representation and the computation directly in code. Together, the fundus and OCT examples highlight the two motivations of MedCode: flexibility to construct question-specific analyses and extensibility to different imaging settings through the same environment, without adding a separate tool implementation for each procedure.

D.2 Cross-Model Visual Comparison

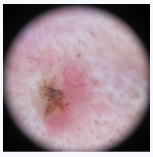
Figure 4 illustrates three ways of approaching the same dermoscopy question: direct visual reasoning with Qwen3-VL-8B, crop-based inspection with DeepEyes-7B, and programmatic analysis with MedCode. The case highlights a central advantage of MedCode: it can construct question-specific evidence that separates competing visual cues, rather than relying on the most salient appearance or repeatedly enlarging the same region.

Beyond Magnification. In the illustrated trajectories, Qwen3-VL-8B emphasizes the irregular brown and gray-black pigmentation and selects melanoma (D). DeepEyes-7B performs two successive crops centered on the dark region. These views expose more local detail, but the interpretation remains dominated by pigment heterogeneity, leading to the same answer. MedCode instead investigates the competing vascular explanation, shifting the analysis from the conspicuous dark core to the peripheral red structures.

Adaptive Evidence Construction. MedCode develops this analysis through three feedback-linked programs. First, it separates excess-red pixels from dark material while excluding the black vignette. The returned masks distinguish the two signals and expose the mixture of diffuse redness and fine structures. Second, it applies a green-channel black-hat filter to suppress the broad background and enhance peripheral lines. The resulting view brings their branching morphology into focus. Third, it computes a top-hat response on the red-minus-green image and intersects the

local responses, checking whether the enhanced lines coincide with locally elevated redness. Each execution thus supplies an intermediate representation that informs the next analytical step.

Integrating Computation with Visual Reasoning. By combining the original crop with the derived masks and overlapping responses, MedCode identifies branching vascular evidence on a pink-white background and selects basal cell carcinoma (A), matching the reference. The distinctive capability is the coupling of flexible computation with iterative interpretation: MedCode specifies which image properties to extract, constructs a targeted check, and uses the returned evidence to resolve the competing explanations. General code therefore expands the reasoning process from selecting an image region to designing the analysis needed for the question.



Question: What is the identified skin abnormality in the provided image?
(A) Basal cell carcinoma
(B) Hives
(C) Seborrheic keratosis
(D) Melanoma

Ground Truth: A

Qwen3-VL-8B

ASSISTANT 1
The lesion contains an irregular brown and gray-black area on a much broader pink-white background. The pigmentation is unevenly distributed rather than forming one uniform patch, and the dark region has an irregular outline. These features make a neoplastic lesion more plausible than hives, which does not explain the focal pigmented structure visible here ....

The strongest alternatives are melanoma and basal cell carcinoma. The pink-white background and visible red lines could fit basal cell carcinoma. However, my attention is drawn to the irregular gray-brown material and the abrupt transition between dark and pale areas. I interpret this color variation as the dominant diagnostic clue, rather than treating the vessels as the defining pattern ... ...

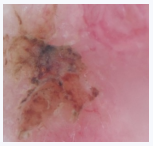
The answer is D. ❌

DeepEyes-7B

ASSISTANT 1
The dark central region is the most suspicious feature. Considering enlarge it to examine the variation in pigmentation and the transition to the surrounding skin.

TOOL CALL 1
{"name": "image_zoom_in_tool", "arguments": {"bbox_2d": [255, 500, 570, 800]}}

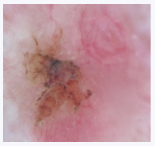
TOOL RETURN 1



ASSISTANT 2
The crop shows heterogeneous brown and gray coloration rather than a uniform deposit. This strengthens my concern for melanoma. Let's widen the crop to check whether the pigmented region has an organized border or a keratotic surface pattern.

TOOL CALL 2
{"name": "image_zoom_in_tool", "arguments": {"bbox_2d": [190, 420, 640, 840]}}

TOOL RETURN 2



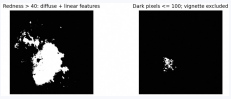
ASSISTANT 3
The wider crop preserves the irregular central dark material against a pale pink field. Although red lines are visible, suggesting that basal cell carcinoma cannot be entirely ruled out, the uneven pigmentation is considered a more significant feature. I retain D, melanoma.
The answer is D. ❌

MedCode

ASSISTANT 1
... ... I will separate excess red signal from dark material, excluding the black vignette.

CODE INTERPRETER 1
```
redness = cv2.subtract(red, green)
red_candidates = (redness > 40) & (fov > 0)
dark_candidates = (gray <= 100) & (fov > 0) … …
```

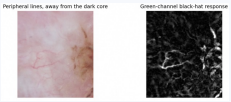
EXECUTION RETURN 1
```
Excess-red candidates: 157,393 px
Dark candidates: 7,128 px
```



ASSISTANT 2
... ... I will suppress the broad background with a green-channel black-hat filter and inspect the peripheral lines away from the dark core.

CODE INTERPRETER 2
```
kernel =
cv2.getStructuringElement(cv2.MORPH_ELLIPS
E, (17, 17))
response = cv2.morphologyEx(green,
cv2.MORPH_BLACKHAT, kernel) … …
```

EXECUTION RETURN 2
```
Peripheral ROI: [100, 470, 315, 715]
Local dark-line candidates: 3,099 px
```



ASSISTANT 3
... ... I will check whether these same locations also have locally enhanced red signal, rather than relying on darkness.

CODE INTERPRETER 3
```
red_response =
cv2.morphologyEx(redness,
cv2.MORPH_TOPHAT,
kernel) … …
```

EXECUTION RETURN 3



ASSISTANT 4
The original peripheral crop and the overlapping responses support branching red vessels on a pink-white background ... ... I choose A. ✅

Figure 4: Illustrative cross-model comparison between Qwen3-VL-8B,DeepEyes-7B, and MedCode.