ABRA: Agent Benchmark for Radiology Applications

Anonymous Authors[1]

Abstract

Existing medical-agent benchmarks deliver imaging as pre-selected samples, never as an environment the agent must navigate. We introduce ABRA, a radiology-agent benchmark in which the agent operates an OHIF viewer and an Orthanc DICOM server through twenty-one function-calling tools that span slice navigation, windowing, series selection, pixel-coordinate annotation, and structured reporting. ABRA contains 655 programmatically generated tasks across three difficulty tiers and eight types (viewer control, metadata QA, vision probe, annotation, longitudinal comparison, BI-RADS reporting, and oracle variants of annotation and BI-RADS reporting), drawn from LIDC-IDRI, Duke Breast Cancer MRI, and NLST New-Lesion LongCT. Each episode is scored along Planning, Execution, and Outcome (Bluethgen et al., 2025) by task-type-specific automatic scorers. Ten current models, five closed-weight and five open-weight, reach at least 89% Execution on real annotation but only 0–25% Outcome; on the paired oracle variant where a simulated detector supplies the finding, Outcome on the same task reaches 69–100% across the models evaluated, localising the bottleneck to perception rather than tool orchestration. Code, task generators, and scorers are released at `https://anonymous.4open.science/r/ABRA-F815`.

1. Introduction

Where benchmarks for vision-language models classically deliver the input as a single prompt (an entire image, a concatenated patient record, a pre-curated case vignette), recent agent benchmarks in software, web, and operating-system domains (Jimenez et al., 2024; Zhou et al., 2024; Xie et al., 2024) present the input as an environment the model queries on demand, making observation itself an action the agent chooses to take. Medical imaging is the exception: the multimodal entries among medical agent benchmarks deliver imaging as fixed samples and never as a queryable environment. Radiology offers a sharp test case because the workstation workflow (load a study, scroll slices, adjust windowing, place annotations, compose a report) is already tool-shaped and standardised over DICOM.

Putting the agent inside a live viewer rather than handing it static images turns four behaviours into measurable signals: pixel-coordinate annotation as a scored action, observation cost (which slices, series, and preprocessor the agent chose to query), scroll and windowing as planning steps, and multi-series navigation across longitudinal studies. None of these are exercised when imaging is delivered as preselected samples, and none are exercised by the one prior radiology-specific agent benchmark, RadABench (Zheng et al., 2024), whose deliberately symbolic design substitutes placeholder tokens for pixels to upper-bound planning and orchestration with perception out of scope. ABRA is the complementary axis on the same problem: to our knowledge, the first benchmark in which a radiology agent's tool use and visual perception are exercised jointly inside a radiology viewer (full comparison in Table 1).

ABRA wraps the OHIF open-source radiology viewer (Ziegler et al., 2020) and an Orthanc DICOM server (Jodogne, 2018) in a controller that drives a headless browser session and exposes the viewer through a uniform function-calling interface. The tool set is partitioned into four observation categories (metadata, viewer screenshot, DICOM pixel, oracle predictions) and three action classes (navigation, segmentation, reporting).

The benchmark contains 655 tasks generated programmatically from three public TCIA cohorts that together cover chest CT (LIDC-IDRI, NLST New-Lesion LongCT) and breast MRI (Duke Breast Cancer MRI), stratified across three difficulty tiers and eight types ranging from viewer control to end-to-end BI-RADS reporting. Perception-heavy types are released in paired *oracle* and *real* variants of the same task: the oracle variant exposes a simulated-detector tool and no pixel-access tools, while the real variant exposes pixel-access tools and no detector. Each episode is scored along Planning, Execution, and Outcome by task-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Do not distribute.

type-specific automatic scorers.

We evaluate ten current models on ABRA, five closed-weight and five open-weight. Annotation Outcome drops from a 0.69–1.00 oracle range to 0.00–0.25 on the real variant across the roster; the gap is the headline finding and is consistent with perception, not tool orchestration, as the binding constraint. The pattern extends beyond the paired tasks: scores are strong on viewer control and metadata retrieval and weaken progressively as a task focuses on direct pixel interpretation. Current models on ABRA are therefore tool-competent but perception-limited on real DICOMs, which suggests the near-term operating point for clinical assistants is one in which a specialised perception model supplies findings and the LLM orchestrates the workflow around them.

Contributions.

- **Environment.** An executable radiology-workstation stack (OHIF viewer, Orthanc PACS, preprocessor) exposed to agents through a uniform function-calling interface, with tools partitioned into four observation categories and three action classes covering the workstation workflow.
- **Benchmark.** 655 programmatically synthesised tasks across eight types and three difficulty tiers drawn from three public TCIA datasets covering chest CT and breast MRI, with paired oracle and real variants of the same task that swap access to a simulated detector for access to pixel-level tools.
- **Evaluation.** Automatic Planning, Execution, and Outcome scorers operating on controller trajectory logs, instantiating the framework of Bluethgen et al. (2025) on real episodes.
- **Empirical findings.** Within-pair gaps between oracle and real variants of the same task across current closed- and open-weight models, consistent with perception as the binding constraint and characterised across task types and difficulty tiers.

2. Related Work

A growing body of agent-benchmark research instantiates its tasks inside real, interactive systems rather than static datasets. Xie et al. (2024) evaluate agents in real operating systems; Zhou et al. (2024) in live Dockerised web stacks; Jimenez et al. (2024) against real GitHub repositories with their native test suites; and Mialon et al. (2024) on real-world tool use across the live web. In medicine, however, interactive clinical benchmarks remain the exception: most medical agent benchmarks run on simulated clinical workflows rather than the imaging environments radiologists actually use.

To structure the comparison, we adopt the four-tier evaluation framework (Planning, Execution, Outcome, and System-level) proposed by Bluethgen et al. (2025) for agentic radiology systems, and position ABRA against the nine prior medical agent benchmarks they catalogue (Table 1).

Most prior work targets text-only clinical decision making or medical question-answering and scores primarily task-success, with limited process or cost evaluation. Three benchmarks (AgentClinic, MedChain, MedAgentBoard) include multimodal imaging inputs, but as static image samples rather than through an interactive imaging environment; none exposes a live viewer or a pixel-level action space.

The closest related benchmark is RadABench (Zheng et al., 2024), the one other radiology-focused agent benchmark with fine-grained per-step agent evaluation. Zheng et al. (2024) deliberately make RadABench a fully symbolic simulator: imaging inputs are placeholder tokens rather than actual pixels, which the authors frame as an "upper-bound" measurement of an agent's planning and orchestration capabilities, with perception out of scope. ABRA is the complement on the same axis: a radiology-specific benchmark that exposes an actual imaging stack (OHIF viewer, Orthanc PACS, DICOM pixels), so tool use and visual perception are exercised jointly. To our knowledge, ABRA is the first medical agent benchmark to place the agent inside a live clinical viewer.

3. ABRA Environment

In this section we describe the problem formulation, introduce the main components of the environment and observation/action spaces.

3.1. Task definition

Following Xie et al. (2024), we cast an ABRA episode as a goal-conditioned partially observable MDP $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \Omega, r, \rho_0, \mathcal{G})$ specialised to a radiology workstation. $\mathcal{S}$ is the joint OHIF viewer and Orthanc PACS state (loaded studies, viewport geometry, placed annotations, DICOM cache). The observation space $\mathcal{O}$ (Section 3.3) is the set of typed return schemas of the read-only tools, and the action space $\mathcal{A}$ (Section 3.4) consists of tool invocations, of which a small terminal subset ends the episode on call. The transition $\mathcal{T}$ and observation function Ω are the deterministic effect of dispatching tool calls through the controller bridge: only the components of $\mathcal{S}$ named by their return schemas enter the next observation, so observation itself is an action the agent chooses to take. The initial-state distribution ρ_0 is degenerate (each task fixes the loaded study, the active series, and the viewport; Section 3.2). A goal $g \in \mathcal{G}$ is the natural-

Table 1. Comparison of ABRA with the nine prior medical agent benchmarks catalogued by Bluethgen et al. (2025) (their Table 2). *Environment*: Static (no interactive env), Chat (dialogue sim), EHR (FHIR/EHR sandbox), Sim (abstract tool simulator), Viewer (live radiology viewer). *Modality*: Text, Multi (text + images). *Tools*: how the agent interacts with the environment.

Benchmark	Environment	Modality	Tools	Reference
General medicine / clinical decision making				
CRAFT-MD	Chat	Text	None	Johri et al. (2025)
AgentClinic	Chat	Multi	Tool-call + RAG	Schmidgall et al. (2025)
MIMIC-CDM	EHR	Text	Tool-call (3)	Bani-Harouni et al. (2026)
MedChain	Chat	Multi	None	Liu et al. (2025)
SDBench	Chat	Text	XML actions	Nori et al. (2025)
MedAgentBench	EHR	Text	9 FHIR calls	Jiang et al. (2025)
MedAgentBoard	Static	Multi	Python exec	Zhu et al. (2025)
MedAgentsBench	Static	Text	None	Tang et al. (2025)
Radiology				
RadABench	Sim	Text	XML tool cards	Zheng et al. (2024)
ABRA	**Viewer**	**Multi**	**21 function-calls**	**this work**

language instruction together with a structured target (final viewport state, free-text string, segmentation contour, list of longitudinal findings, or BI-RADS report). The reward r is the per-episode composite $S = 0.20\,P + 0.30\,E + 0.50\,O$ defined in Section 4.3, applied once at termination.

Episode dynamics. At each turn the agent receives the conversation so far and emits a (possibly empty) batch of tool calls from $\mathcal{A}$; the controller dispatches each call, returns its typed result as a separate observation, and re-invokes the agent. Initial context is the instruction plus a snapshot of the reset viewport, with no DICOM tags, pixels, or oracle predictions pre-loaded. Two properties are distinctive. Observations are pull-based, which makes tool selection itself a measurable component of the score (Section 4.3). Termination is task-shaped, with each task type carrying its own submission action (`submit_birads_report`, `submit_longitudinal_complete`, `submit_answer`) rather than a global `DONE` signal; an episode ends when any of these is invoked, when the agent returns an empty batch with a final text response, or when the agent reaches the task's cap on the number of turns, after which the controller freezes the trajectory and records a final viewport snapshot for the three scorers. If the episode terminates without the task's submission action being invoked, the outcome scorer returns zero for that episode, since no scorable deliverable was produced. Each task fixes this turn cap at generation time; a separate per-response output-token cap is a property of the agent configuration rather than the task, and is reported with the experimental setup in Section 5.

3.2. Viewer as agent environment

ABRA is an executable and resettable environment built around the OHIF open-source radiological viewer (Ziegler et al., 2020) paired with an Orthanc DICOM server (Jodogne, 2018), mirroring the PACS-plus-viewer stack used in clinical workstations so that the agent workflow (loading a study, scrolling slices, adjusting windowing, placing annotations, writing a report) remains the one radiologists themselves follow. A dedicated OHIF extension surfaces the viewer's internal state (viewport geometry, active series, placed annotations) and the built-in segmentation and measurement modules, while an out-of-process preprocessor converts raw DICOM pixels into model-appropriate PNGs whose coordinate frame is shared with the segmentation actions. ABRA supports two interaction modes over the same underlying tool set: the in-browser chat extension (Figure 3) executes tools directly inside a user-driven OHIF session, and the benchmark controller drives a headless OHIF instance through Puppeteer and exposes the same tools over an HTTP interface (Figure 1), so every evaluated agent operates against identical viewer state. Episodes run in isolated browser contexts, which makes them cheap to parallelise and trivially resettable between runs.

Initialization. At the start of each episode the controller instructs the Orthanc server to load the study referenced in the task specification, clears any persisted viewer state (active series, windowing presets, placed annotations), and positions the viewport on the default series and slice for that task type. It then assembles the initial context delivered to the agent: the natural-language instruction, the system

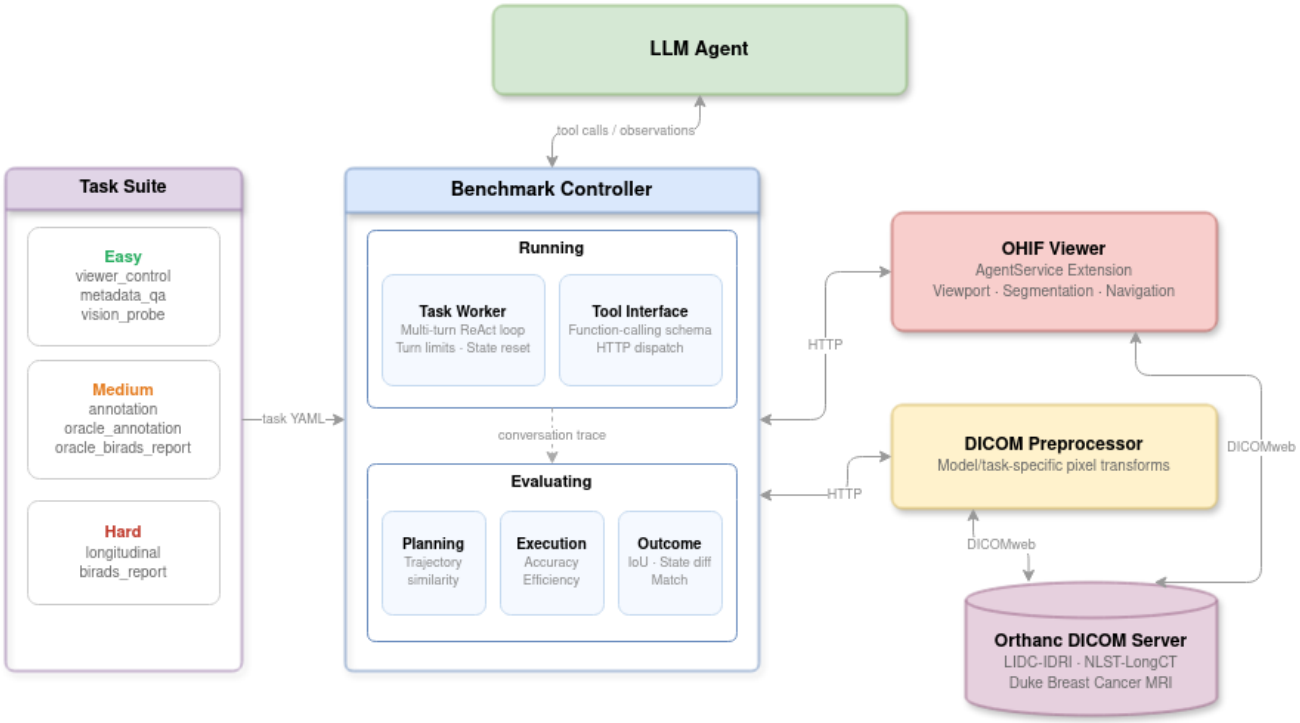

Figure 1. Overview of the ABRA architecture. The benchmark controller draws tasks from the suite, drives an LLM agent through a multi-turn loop, and dispatches its tool calls to a headless OHIF viewer and a DICOM preprocessor, both backed by an Orthanc PACS populated with three TCIA datasets. After the agent terminates, the controller scores the trajectory along Planning, Execution, and Outcome. Episodes run in isolated browser contexts and are trivially parallelisable.

prompt (Appendix B.2), and a snapshot of the reset viewport. No pixel data, DICOM tags, or oracle predictions are pre-loaded; every subsequent observation must be fetched through an explicit tool call. Further details are given in Appendix B.3.

Evaluation. Once the agent emits a terminal action (a report submission, a free-text answer, or the longitudinal completion signal), or its step budget is exhausted, the controller freezes the viewer and records three artefacts: the ordered sequence of tool calls, per-call return values and execution metadata, and the final structured deliverable. These feed the three scorers of our evaluation framework (Planning, Execution, and Outcome), whose benchmark-specific formulations are presented in Section 4.3, with full per-task-type breakdowns in Appendix B.4. Browser state is then discarded before the next episode begins.

3.3. Observation space

The observation space $\mathcal{O}$ of ABRA is structured as a set of read-only tools that the agent queries on demand. Only the natural-language task instruction and a snapshot of the current viewport state are delivered in the initial prompt; all other information (DICOM tags, pixel content, simulated external-model outputs) must be fetched explicitly through tool calls. This design bounds token consumption on long-horizon episodes and makes tool-call planning a measurable part of the Planning score (Sec. 4.3). We group observations into four categories: (i) *Metadata*, six query tools that expose DICOM tags at study, series, and instance granularities, plus the live viewport state and loaded segmentations; (ii) *Viewer screenshot*, a full-resolution rendering of the OHIF browser UI captured through Puppeteer, intended for inspecting overlays, toolbar state, and already-placed annotations; (iii) *DICOM pixel*, a preprocessor that routes a requested slice through one of six named pipelines (`default`, `lung_window`, `soft_tissue_window`, `percentile_norm`, `breast_mri`, `raw_uint16`) and returns a PNG whose coordinate frame is shared with the segmentation actions in Sec. 3.4; and (iv) *Oracle predictions*, two tools that simulate calls to external CAD models and return structured findings, which lets us decouple tool-use competence from visual perception. Full tool signatures, return schemas, and preprocessor definitions are given in Appendix A.2.

3.4. Action space

Actions in ABRA fall into three classes. *Navigation and display* actions modify the active viewport and therefore change what the agent will next observe. *Segmentation*

actions deposit region-of-interest annotations in pixel coordinates; providing circle and bounding-box primitives alongside arbitrary polygons matches the shapes used in typical radiology workflows and keeps the interaction surface narrow without losing the capabilities we want to measure. *Reporting* actions produce the structured deliverable for the task (BI-RADS report, longitudinal finding, free-text answer). Most reporting actions are *terminal*, ending the episode on invocation and handing control to the outcome scorer. The per-episode tool set is scoped by task type (for example, viewer-control tasks see only the navigation class), so every episode evaluates both the agent's ability to select a correct tool and its ability to parameterise it correctly. Pixel coordinates are returned by `get_dicom_image` and accepted by every segmentation action in the same frame, so they can be round-tripped between observations and actions without an explicit transformation. The complete action specification is provided in Appendix A.3.

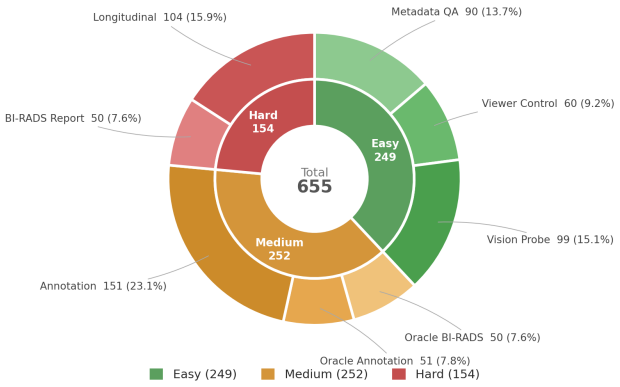

Figure 2. Task distribution by difficulty and type for ABRA.

4. ABRA Benchmark

4.1. Task generation

ABRA tasks are synthesised programmatically from three public TCIA cohorts (Clark et al., 2013), stratified over three difficulty tiers, and instantiated from one of eight task types that span the radiologist's workstation workflow. Each generated task contains a natural-language instruction, an initial viewer state, a ground-truth target, and a reference tool-call trajectory used by the planning scorer. We provide 655 tasks across the three source cohorts; their joint distribution over types and difficulties is shown in Figure 2.

Datasets. The three datasets were selected to exercise distinct diagnostic scenarios through a shared acquisition pipeline. *LIDC-IDRI* (Armato et al., 2011) contributes thoracic CT studies with per-nodule contours from up to four radiologists, aggregated into consensus segmentations using the 50% volumetric majority-vote convention; these anchor the annotation-heavy middle tier. *Duke Breast Cancer MRI* (Saha et al., 2021) contributes multi-sequence breast MRI studies with BI-RADS labels and drives the structured-reporting tasks. *NLST New-Lesion LongCT* (Gong et al., 2025) contributes curated baseline-and-follow-up CT pairs with new-lesion annotations and drives the longitudinal comparison tasks. Per-dataset cohort sizes, modality counts, and task contributions are detailed in Appendix B.1.

Difficulty tiers. Tasks are partitioned into three tiers whose composition lets us probe different capability profiles. The *Easy* tier contains short viewer-control, metadata, and modality-recognition tasks: either a simple tool call against the viewer or DICOM metadata, or a single-image judgement about the study's modality and appropriate preprocessor. The *Medium* tier splits along a deliberate axis: some tasks require the agent to perceive anatomy directly on CT and place a region-of-interest contour, while the oracle variants hand the model structured findings from a simulated detector and ask it to act on them, separating reasoning and tool orchestration from visual perception. The *Hard* tier targets realistic clinical workflows end-to-end: longitudinal comparison between a baseline and a follow-up CT, and structured BI-RADS reporting on a full multi-sequence breast MRI.

Task types. The eight task types are summarised in Table 2; prompt templates and the per-type tool-availability matrix are given in Appendix B.2.

4.2. Tools

The twenty-one tools that together form the observation and action spaces were introduced in Sections 3.3 and 3.4; we briefly describe how they are surfaced to the agent. Each tool is presented as a typed JSON function-calling schema (in the form introduced by OpenAI), and the controller translates every schema invocation into an HTTP request against the benchmark bridge described in Section 3.2. The per-task-type tool visibility set (detailed in Appendix B.2) is injected into the agent's initial prompt, so an agent never sees a tool that is irrelevant to its current task.

4.3. Evaluation

ABRA scoring instantiates three of the four tiers in the framework that Bluethgen et al. (2025) propose for evaluating agentic radiology systems (their Figure 4): Planning, Execution, and Outcome. Where that work defines the tiers conceptually, we instantiate each with automatic scorers that run on the controller's trajectory logs without any human-in-the-loop review. Every episode yields a composite score $S = 0.20\,P + 0.30\,E + 0.50\,O$, with P, E, O the Planning, Execution, and Outcome scores in $[0, 1]$. The weighting reflects that task success is the end goal, while

Table 2. The eight ABRA task types, grouped by difficulty tier.

Tier	Type	Description
Easy	`viewer_control`	Viewport manipulation: slice navigation, window/level presets, series switching.
	`metadata_qa`	DICOM tag retrieval at study, series, and instance granularity.
	`vision_probe`	Modality recognition and preprocessor selection from a single rendered image.
Medium	`annotation`	Perceive pathology on a CT series and contour it with a segmentation primitive.
	`oracle_annotation`	Same target as `annotation`, with oracle detector findings supplied.
	`oracle_birads_report`	Compose a BI-RADS report from oracle breast-MRI findings.
Hard	`longitudinal`	Compare a baseline-and-follow-up CT pair and submit each new lesion with its slice and pixel location.
	`birads_report`	Read a full multi-sequence breast MRI and produce an end-to-end BI-RADS report.

Planning and Execution interpret *how* success (or failure) was reached. Full formulas for every component are given in Appendix B.4; the subsections below summarise the main design choices.

Planning. We compare the agent's ordered trajectory T_{agent} against a reference trajectory T_{ref} attached to each task at generation time, collapsing both to unordered tool multisets and scoring F1 between them. Unordered matching was chosen because multiple tool orderings correspond to valid radiological workflows (for example, reviewing metadata before or after scrolling through slices), and penalising order would encode a single canonical sequence. Extraneous calls not present in T_{ref} incur a small redundancy penalty (0.05 per extra call, capped at 0.30), which discourages pathological "call every tool" behaviour without erasing credit for legitimate exploration.

Design choice and limitation. Reference trajectories are generated programmatically from task templates. This is a best-effort estimate of the reference plan rather than an oracle ground truth. The high Planning and Execution scores on the easy and medium tiers in Table 4 are consistent with the programmatic references capturing plausible solution routes when the workflow is short and well-constrained, and their divergence on the hard tier marks the regime where independent validation would be most informative. A detailed discussion, together with a plan to collect reference trajectories from real radiologist sessions, is given in Appendix D.

Execution. Execution assesses the agent step-by-step inside the Reason-Act-Observe loop. Four components are aggregated into a weighted sum $E = 0.40\, A_{\text{tool}} + 0.20\, Q_{\text{param}} + 0.25\, E_{\text{turn}} + 0.15\, R_{\text{err}}$, where A_{tool} is the fraction of tool calls that returned success, Q_{param} the semantic quality of the arguments passed to each call, E_{turn} turn efficiency relative to the reference trajectory length, and R_{err} whether the agent recovered gracefully from a tool error instead of looping. Tool-call success carries the largest weight because failed calls (whether from wrong tool selection or malformed arguments) most directly cascade into bad outcomes. Parameter quality is weighted more lightly because most type-level errors are already caught by the function-calling schema, so what remains to score is semantic quality (selecting an appropriate preprocessor, targeting the correct series UID). Turn efficiency penalises aimless long trajectories; error recovery rewards the rarer but valuable behaviour of turning a tool failure into a correct next action.

Outcome. Outcome evaluates whether the task was completed correctly, independent of how the agent got there. Because the eight task types produce structurally different deliverables (numeric viewer state, free-text answer, segmentation contour, structured BI-RADS report, list of longitudinal findings), we use task-type-specific scorers rather than a single metric, summarised in Table 3. Full scorer definitions, their input formats, and the BI-RADS field-weight rationale are given in Appendix B.4.3.

5. Experimental Evaluation

We evaluate ten model checkpoints across two access regimes. Five are vendor-locked frontier APIs (Claude Sonnet 4.6 (Anthropic, 2026); GPT-5.4 and GPT-5.4-nano (Singh et al., 2026); Gemini 3 Flash (Google DeepMind, 2025b); Gemini 3 Pro (Google DeepMind, 2025a)); five are open-weights checkpoints accessed via Ollama Cloud (Gemma 4 (Google DeepMind, 2026); Qwen 3.5 (Team, 2026); Ministral 3 14B (Liu et al., 2026); Mistral Large 3 (Mistral AI, 2025); Kimi K2.5 (Team et al., 2026)). Each task's turn cap is fixed at generation time (Section 3.1); we additionally cap output at 20,048 tokens per response, decode at temperature 0 with provider-default reasoning effort (medium across all evaluated models), and report a

Table 3. ABRA outcome scorers, listed by task type. All scores are normalised to $[0, 1]$ so the composite remains bounded.

Scorer	Task type(s)	Definition
state-diff	`viewer_control`	Exact match of the final viewport state against the target.
exact-match	`metadata_qa`, `vision_probe`	String-level match of the submitted answer.
IoU	`annotation`, `oracle_annotation`	Intersection-over-union against the consensus contour, divided by the IoU of the best-fit primitive of the agent's chosen shape so a correctly placed circle is not penalised against a polygon.
point-distance	`longitudinal` (single-lesion)	Pixel distance between the submitted centre and the ground-truth lesion centre.
multi-finding	`longitudinal` (multi-lesion)	Detection rate minus 0.1 per false positive, clamped to $[0, 1]$, with the same slice-and-distance match criterion as the single-lesion scorer.
BI-RADS-field	`birads_report`, `oracle_birads_report`	Weighted average of per-field agreement (laterality, category, lesion count, enhancement, quadrant).

single run per (model, task) pair. Anthropic prompt caching is invoked explicitly on the system block and a trailing breakpoint over the growing conversation, while OpenAI and Gemini rely on the providers' automatic prefix caching. Tasks that exit on the turn cap without invoking the relevant submit action receive Outcome $= 0$.

Table 18 reports per-model token consumption and serialised runtime across the full benchmark, broken down by difficulty tier. Per-task-type breakdowns with prompt-cache hit ratios are given in Appendix F.

We observe that Outcome scores broadly follow the difficulty grading: Easy-tier Outcome stays in the upper range, Medium drops substantially, and Hard converges to near-floor values. Both the Medium and Hard tier averages, however, mask a non-uniform contribution from their constituent task types. On Medium, annotation without oracle conditioning largely fails, while the oracle-conditioned variants mostly succeed and account for most of the achieved Medium score. On Hard, the achieved score is dominated by BI-RADS reporting, which admits competitive scores from prior-based guessing on a well-known categorical distribution and earns partial credit on off-by-one ordinal errors; longitudinal new-lesion detection admits no comparable shortcut (per-task-type breakdown in Appendix C.1). The Execution sub-score, by contrast, sits near its ceiling on every tier and for most models (Execution ≈ 1.0): tool calls succeed, function arguments are correctly typed, and trajectories close within the per-task turn budget at near-unit rates.

5.1. Analysis

Vision is the dominant bottleneck. Across task types, the location and depth of failures point to vision as the dominant constraint. Annotation without oracle conditioning sits near the floor of the Outcome scale on every model in the roster, and the residual Outcome on Hard concentrates in the BI-RADS task type where prior-based guessing is available; the longitudinal new-lesion task, which requires actually localising imaging findings on follow-up volumes, remains at near-zero across the roster (per-task-type breakdown in Appendix C.1). One observation in the opposite direction is worth noting: the Gemini family (both Flash and Pro) produces the only annotation results in the roster that score meaningfully above chance on the more pronounced LIDC-IDRI nodules.

Tool-call mechanics. Execution sub-scores sit at the top of their range across the entire roster (Table 4); Planning is high on Easy and Medium but drops on Hard, a limitation of the synthetic reference trajectories (Appendix D). This highlights that the capability gap of current models is mostly clinical: agents drive the viewer, query DICOM tags, and invoke submission actions along the trajectories the benchmark expects, with rough congruence to the reference trajectories used by the Planning scorer. At the same time, models occasionally omit findings on multi-lesion longitudinal tasks, and the smaller checkpoints sometimes alter oracle annotations that should pass through unchanged (oracle-alteration rates per model in Table 15). Clinical reliability, rather than tool-call mechanics, appears to be the tighter constraint on Outcome.

6. Conclusion

ABRA is an agent benchmark that places the model inside a live radiology workstation: an OHIF viewer and Orthanc PACS driven through a controller bridge that exposes function-calling tools spanning observation, navigation, segmentation, and structured reporting. On top of this environment we release 655 programmatically synthesised tasks across eight types and three difficulty tiers, drawn from three public TCIA cohorts, with paired oracle and real variants that isolate visual perception from tool orchestration; trajectories are scored along Planning, Execution, and Outcome (Bluethgen et al., 2025). Empirically, frontier models

Table 4. ABRA evaluation across difficulty tiers. Per-tier scores are Planning (P), Execution (E), Outcome (O), and composite average $S = 0.20P + 0.30E + 0.50O$. The final column reports the n-weighted average across all eight task types. Higher is better.

	Easy				Medium				Hard				Overall
Model	P	E	O	**Avg**	P	E	O	**Avg**	P	E	O	**Avg**	
Claude Sonnet 4.6	0.93	0.99	0.86	0.91	0.78	**0.99**	0.41	0.66	0.20	0.98	**0.21**	0.44	**0.70**
GPT-5.4	0.83	0.99	0.88	0.91	**0.82**	**0.99**	0.42	**0.67**	0.31	0.95	0.11	0.40	**0.70**
Qwen 3.5	0.88	**1.00**	**0.91**	**0.93**	0.76	0.93	0.39	0.63	0.49	**0.99**	0.11	0.45	**0.70**
GPT-5.4-nano	**0.95**	0.99	0.87	0.92	0.71	**0.99**	0.39	0.64	**0.56**	0.97	0.04	0.42	0.69
Gemma 4	0.88	**1.00**	0.87	0.91	0.75	0.93	0.43	0.64	0.38	0.93	0.02	0.37	0.68
Mistral Large 3	0.90	0.98	0.69	0.82	0.80	**0.99**	0.38	0.65	0.51	0.90	0.15	0.45	0.67
Gemini 3 Flash	0.61	0.93	0.88	0.84	0.58	0.95	**0.53**	**0.67**	0.37	0.86	0.06	0.36	0.66
Ministral 3 (14B)	0.92	0.99	0.68	0.82	0.78	0.98	0.38	0.64	0.43	0.87	0.12	0.41	0.66
Gemini 3 Pro	0.62	0.94	0.79	0.80	0.62	0.98	0.35	0.59	0.33	0.96	0.16	0.44	0.64
Kimi K2.5	0.73	0.89	0.75	0.79	0.68	0.98	0.37	0.61	0.44	0.98	0.14	**0.46**	0.64

API snapshots: GPT-5.4 (`gpt-5.4-2026-03-05`), GPT-5.4-nano (`gpt-5.4-nano-2026-03-17`), Gemini 3 Flash (`google/gemini-3-flash-preview-20251217`), Gemini 3 Pro (`google/gemini-3.1-pro-preview-20260219`).

Table 5. Outcome and overall Avg on the four paired (oracle vs *real*) task variants: annotation and BI-RADS reporting.

Model	Oracle				Real			
	Annotation		BI-RADS		Annotation		BI-RADS	
	Out ↑	**Avg** ↑	Out ↑	**Avg** ↑	Out ↑	**Avg** ↑	Out ↑	**Avg** ↑
Claude Sonnet 4.6	**1.00**	**0.98**	**1.00**	**0.96**	0.02	0.45	**0.64**	**0.73**
GPT-5.4	0.98	0.97	**1.00**	**0.96**	0.03	0.47	0.32	0.53
Mistral Large 3	0.91	0.93	**1.00**	**0.96**	0.00	0.45	0.47	0.60
Gemini 3 Flash	0.90	0.82	**1.00**	0.93	**0.25**	**0.53**	0.18	0.41
Qwen 3.5	0.95	0.94	**1.00**	**0.96**	0.00	0.42	0.35	0.59
Kimi K2.5	0.84	0.80	0.94	0.93	0.02	0.45	0.42	0.62
Ministral 3 (14B)	0.90	0.91	**1.00**	**0.96**	0.00	0.45	0.35	0.46
GPT-5.4-nano	0.96	0.94	**1.00**	**0.96**	0.00	0.42	0.12	0.45
Gemini 3 Pro	0.69	0.73	0.58	0.71	0.16	0.51	0.50	0.65
Gemma 4	0.88	0.87	**1.00**	**0.96**	0.08	0.46	0.06	0.33

drive the viewer and compose structured deliverables at near-ceiling Execution, but Outcome on real annotation variants falls to 0.00–0.25 while paired oracle variants recover to 0.69–1.00, localising the gap to perception rather than tool-call mechanics on the current generation. We position ABRA as a measurement substrate for that gap rather than a verdict on clinical readiness; benchmark limitations and the future directions they motivate are discussed in Appendix D.

Acknowledgements

Do not include acknowledgements in the initial version of the paper submitted for blind review.

Impact Statement

This paper introduces a benchmark for evaluating AI agents on radiological imaging tasks. Its purpose is measurement rather than deployment: we release tools, tasks, and scorers so that progress on agentic radiology workflows can be tracked. Benchmark results in this paper should not be read as evidence that current models are clinically ready; the central finding is the opposite, that perception on real DICOMs is the dominant bottleneck at present. Clinical use of any of the evaluated models would require independent regulatory validation, prospective studies on the target population, and integration with workflows beyond the proxies measured here. The data sources (LIDC-IDRI, Duke Breast Cancer MRI, NLST) are public, de-identified, and governed by the licenses listed in Appendix B.1; we add no new patient data. Aside from these standard considerations, there are many potential societal consequences of advancing agentic medical AI, none of which we feel must be specifically highlighted here.

References

Anthropic. Claude Sonnet 4.6 system card. https://www.anthropic.com/claude-sonnet-4-6-system-card, 2026.

Armato, 3rd, S. G., McLennan, G., Bidaut, L., McNitt-Gray, M. F., Meyer, C. R., Reeves, A. P., Zhao, B., Aberle, D. R., Henschke, C. I., Hoffman, E. A., Kazerooni, E. A., MacMahon, H., Van Beeke, E. J. R., Yankelevitz, D., Biancardi, A. M., Bland, P. H., Brown, M. S., Engelmann, R. M., Laderach, G. E., Max, D., Pais, R. C., Qing, D. P. Y., Roberts, R. Y., Smith, A. R., Starkey, A., Batrah, P., Caligiuri, P., Farooqi, A., Gladish, G. W., Jude, C. M., Munden, R. F., Petkovska, I., Quint, L. E., Schwartz, L. H., Sundaram, B., Dodd, L. E., Fenimore, C., Gur, D., Petrick, N., Freymann, J., Kirby, J., Hughes, B., Casteele, A. V., Gupte, S., Sallamm, M., Heath, M. D., Kuhn, M. H., Dharaiya, E., Burns, R., Fryd, D. S., Salganicoff, M., Anand, V., Shreter, U., Vastagh, S., and Croft, B. Y. The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans. *Med. Phys.*, 38(2): 915–931, February 2011.

Bani-Harouni, D., Pellegrini, C., Özsoy, E., Navab, N., and Keicher, M. Language agents for hypothesis-driven clinical decision making with reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7vHUQCMAzG.

Bluethgen, C., Veen, D. V., Truhn, D., Kather, J. N., Moor, M., Polacin, M., Chaudhari, A., Frauenfelder, T., Langlotz, C. P., Krauthammer, M., and Nooralahzadeh, F. Agentic systems in radiology: Design, applications, evaluation, and challenges, 2025. URL https://arxiv.org/abs/2510.09404.

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., Phillips, S., Maffitt, D., Pringle, M., Tarbox, L., and Prior, F. The Cancer Imaging Archive (TCIA): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, 2013. doi: 10.1007/s10278-013-9622-7.

Gong, A., Daly, M., Goldin, J., Brown, M., McNItt-Gray, M., and Ruchalski, K. New lung lesions in low-dose ct: a newly annotated longitudinal dataset derived from the national lung screening trial, 2025. URL https://www.cancerimagingarchive.net/analysis-result/nlst-new-lesion-longct/.

Google DeepMind. Gemini 3 Pro. https://deepmind.google/models/gemini/pro/, 2025a. Gemini 3 Pro model page.

Google DeepMind. Introducing Gemini 3 Flash: Frontier intelligence built for speed. https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/, December 2025b. Released December 17, 2025.

Google DeepMind. Gemma 4: Byte for byte, the most capable open models. https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/, April 2026. Open-weights release; model card at https://ai.google.dev/gemma/docs/core/model_card_4.

Jiang, Y., Black, K. C., Geng, G., Park, D., Zou, J., Ng, A. Y., and Chen, J. H. Medagentbench: A virtual ehr environment to benchmark medical llm agents. *NEJM AI*, 2(9), August 2025. ISSN 2836-9386. doi: 10.1056/AIdbp2500144. URL http://dx.doi.org/10.1056/AIdbp2500144.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. R. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Jodogne, S. The Orthanc ecosystem for medical imaging. *Journal of Digital Imaging*, 31(3):341–352, Jun 2018. ISSN 1618-727X. doi: 10.1007/s10278-018-0082-y. URL https://doi.org/10.1007/s10278-018-0082-y.

Johri, S., Jeong, J., Tran, B. A., Schlessinger, D. I., Wongvibulsin, S., Barnes, L. A., Zhou, H.-Y., Cai, Z. R., Van Allen, E. M., Kim, D., Daneshjou, R., and Rajpurkar, P. An evaluation framework for clinical use of large language models in patient interaction tasks. *Nature Medicine*, 31(1):77–86, 2025. ISSN 1546-170X. doi: 10.1038/s41591-024-03328-5. URL https://doi.org/10.1038/s41591-024-03328-5.

Liu, A. H., Khandelwal, K., Subramanian, S., Jouault, V., Rastogi, A., Sadé, A., et al. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

Liu, J., Wang, W., Ma, Z., Huang, G., Su, Y., Chang, K.-J., Li, H., Shen, L., Lyu, M., and Chen, W. Medchain: Bridging the gap between LLM agents and clinical practice through interactive sequential benchmarking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=YvuufwkFJY.

Mialon, G., Fourrier, C., Wolf, T., LeCun, Y., and Scialom, T. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Mistral AI. Mistral Large 3. https://docs.mistral.ai/models/mistral-large-3-25-12, 2025.

Nori, H., Daswani, M., Kelly, C., Lundberg, S., Ribeiro, M. T., Wilson, M., Liu, X., Sounderajah, V., Carlson, J., Lungren, M. P., Gross, B., Hames, P., Suleyman, M., King, D., and Horvitz, E. Sequential diagnosis with language models, 2025. URL https://arxiv.org/abs/2506.22405.

Saha, A., Harowicz, M. R., Grimm, L. J., Weng, J., Cain, E. H., Kim, C. E., Ghate, S. V., Walsh, R., and Mazurowski, M. A. Dynamic contrast-enhanced magnetic resonance images of breast cancer patients with tumor locations, 2021. URL https://www.cancerimagingarchive.net/collection/duke-breast-cancer-mri/.

Schmidgall, S., Ziaei, R., Harris, C. W., Kim, J. W., Reis, E. P., Jopling, J. K., and Moor, M. Agentclinic: a multimodal agent benchmark to evaluate AI in simulated clinical environments, 2025. URL https://openreview.net/forum?id=ak7r4He1qH.

Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., et al. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Tang, X., Shao, D., Sohn, J., Chen, J., Zhang, J., Xiang, J., Wu, F., Zhao, Y., Wu, C., Shi, W., Cohan, A., and Gerstein, M. Medagentsbench: Benchmarking thinking models and agent frameworks for complex medical reasoning, 2025. URL https://arxiv.org/abs/2503.07459.

Team, K., Bai, T., Bai, Y., Bao, Y., Cai, S. H., Cao, Y., et al. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Team, Q. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=tN61DTr4Ed.

Zheng, Q., Wu, C., Qiu, P., Dai, L., Zhang, Y., Wang, Y., and Xie, W. How well can modern LLMs act as agent cores in radiology environments?, 2024. URL https://arxiv.org/abs/2412.09529.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Zhu, Y., He, Z., Hu, H., Zheng, X., Zhang, X., Wang, Z., Gao, J., Ma, L., and Yu, L. Medagentboard: Benchmarking multi-agent collaboration with conventional methods for diverse medical tasks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=BPpG4qQaNj.

Ziegler, E., Urban, T., Brown, D., Petts, J., Pieper, S. D., Lewis, R., Hafey, C., and Harris, G. J. Open health imaging foundation viewer: An extensible open-source framework for building web-based imaging applications to support cancer research. *JCO Clin. Cancer Inform.*, 4 (4):336–345, April 2020.

A. Details of ABRA Environment

A.1. Environment Infrastructure

The ABRA environment is composed of three services that together reproduce a clinical workstation: an Orthanc DICOM server acting as the PACS, an OHIF viewer (v3.9.0) fronted by a custom extension that surfaces internal OHIF tools for model usage, and a Python preprocessing sidecar that handles all DICOM pixel work.

Preprocessor sidecar. DICOM pixel work is handled outside the viewer. The sidecar pulls the requested instance from Orthanc via WADO-RS, runs one of the six named pipelines (Section 3.3) that cover modality-appropriate windowing and model-specific input formatting, and returns a base64 PNG anchored to the coordinate frame used by the segmentation actions. Decoupling preprocessing from the viewer keeps pixel transformations deterministic and independent of viewer rendering state.

User-facing interface. The viewer includes a chat panel extension (Figure 3) that surfaces the same tool set as the benchmark, connected to a model endpoint of the user's choice.

Benchmark automation. For automated evaluation, a separate Node bridge drives a headless instance of this environment, mapping each HTTP request from the benchmark controller to a typed entry point inside the OHIF page; the same tool surface is therefore exercised whether a user or an LLM is at the other end. A parallel evaluation mode replicates the viewer and bridge into multiple independent instances running against the shared Orthanc PACS, with the controller dispatching one task to each free instance.

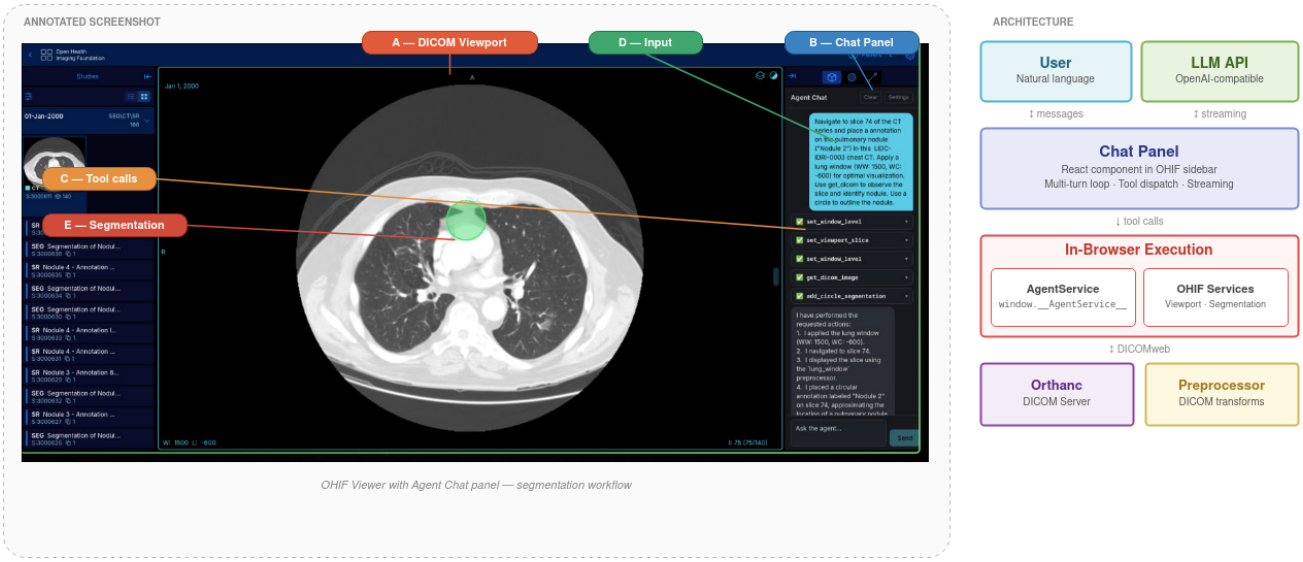

Figure 3. Agent execution path inside OHIF. A user prompt (D) submitted via the embedded chat panel (B) drives streamed tool calls (C) executed in-browser against the viewport (A) and segmentation layer (E); DICOM data is stored in Orthanc, and pixel observations reach the agent through a model-specific preprocessor.

A.2. Observation Space

ABRA exposes observations as a set of read-only tools the agent invokes on demand. Only the natural-language task instruction is delivered in the initial prompt; all other state (such as study headers, slice pixels, viewport settings, and external-model outputs) must be queried explicitly, which bounds token usage on long-horizon episodes and makes tool-call planning an important part of scoring. All pixel observations share a single coordinate convention: the (x, y) frame returned by `get_dicom_image` matches the frame expected by every segmentation action in A.3. We implement four observation categories: **metadata**, **viewer screenshot**, **DICOM pixel**, and **oracle predictions**.

A.2.1. METADATA

Six read-only query tools return DICOM tags at three granularities, the current viewport state, and loaded annotations:

- `get_study_metadata(study_uid)`: study-level tags (`StudyDate`, `PatientID`, `PatientName`, `Modality`, `StudyDescription`) plus a summary list of series (UID, description, modality, series number, instance count).
- `get_study_series(study_uid)`: same series list with up to three sample instances per series, enabling `SOPInstanceUID` selection without paging through the full study.
- `get_series_metadata(series_uid)`: geometry (`SliceThickness`, `PixelSpacing`, `ImageOrientationPatient`, `Rows`, `Columns`) and body-part information for a single series.
- `get_instance_metadata(study_uid, series_uid, sop_uid)`: full DICOM tag set for a single SOP instance, including windowing defaults (`WindowCenter`, `WindowWidth`), rescale parameters (`RescaleSlope`, `RescaleIntercept`), and geometric positioning (`ImagePositionPatient`).
- `get_viewport_state()`: current viewport: slice index, total image count, window width/centre, zoom, active `SeriesInstanceUID`, and loaded `displaySetInstanceUIDs`.
- `list_segmentations()`: all segmentations currently loaded in the viewer, with identifiers, labels, and per-segment details.

A.2.2. VIEWER SCREENSHOT

`get_viewer_screenshot()` captures the full OHIF browser UI as a base64-encoded PNG through Puppeteer. The screenshot includes all on-screen overlays (hanging protocol layout, toolbar state, and any annotations the agent has already placed), allowing the model to inspect the UI directly. For medical image interpretation we provide `get_dicom_image` instead, which allows for model-specific preprocessing and high-fidelity input for vision-based tasks.

A.2.3. DICOM PIXEL

DICOM pixel values are 12–16-bit and modality-dependent, so a raw array is rarely usable by a vision–language model without windowing and channel rearrangement. `get_dicom_image(study_uid, series_uid, slice_index, preprocessor)` routes a requested slice through a named preprocessing pipeline in the `preprocessor` sidecar and returns a base64 PNG whose pixel dimensions match the coordinate frame expected by the segmentation actions. We implement six pipelines: `default` (modality-aware default windowing), `lung_window`, `soft_tissue_window`, `percentile_norm`, `breast_mri`, and `raw_uint16`. Each task description names the expected preprocessor, and the execution scorer penalises modality-incompatible choices (e.g., a CT preset applied to an MR study).

A.2.4. ORACLE PREDICTIONS

To decouple tool-use competence from visual perception, we expose two oracle tools that simulate calls to external CAD models:

- `query_pathology_model(series_uid, slice_index?)` simulates a nodule detector on a CT series. Without `slice_index` it returns an overview of all detected findings (labels, slice ranges, confidence scores, representative slices). With a specific `slice_index` it returns the polygon contour on that slice in pixel coordinates, directly compatible with `add_polygon_segmentation`.
- `query_birads_model(series_uid)` simulates a breast-MRI CAD model and returns structured BI-RADS findings: laterality, lesion count, BI-RADS assessment category, enhancement status, and per-lesion quadrant where available.

A.3. Action Space

Actions in ABRA fall into three classes: **navigation and display**, **segmentation**, and **reporting**. A subset of actions is *terminal*: invoking one ends the episode and hands control to the outcome scorer. Task-type-specific tool sets restrict which classes an agent can use (e.g., viewer-control tasks see only navigation tools). Segmentation and longitudinal actions take pixel-space coordinates in the frame returned by `get_dicom_image`.

A.3.1. Navigation and Display

Four tools modify the active viewport; their effect is observed via `get_viewport_state` or `get_viewer_screenshot`.

- `set_viewport_slice(slice_index)` navigates to a 0-based slice index.
- `set_window_level(window_width, window_center)` sets the display window width and centre; the values are passed straight to OHIF's windowing controls, so the accepted range depends on the modality of the loaded series.
- `set_zoom(scale)` sets the viewport zoom factor; smaller values zoom in.
- `select_series(series_uid)` switches the active viewport to a different series within the loaded study.

A.3.2. Segmentation

ABRA exposes three annotation primitives that cover the ROI shapes used in typical radiology workflows. Providing circle and bounding-box tools alongside the general polygon simplifies the interaction surface while still capturing the capabilities we want to measure: a VLM's ability to perceive the relevant structures and apply medical-domain reasoning to image coordinates. The IoU scorer normalises the agent's IoU against the best-fit shape of the chosen type, so a correctly placed circle or rectangle is not penalised relative to a polygon.

- `add_circle_segmentation(label, slice_index, center, radius)` places a circle in pixel coordinates.
- `add_rectangle_segmentation(label, slice_index, top_left, bottom_right)` places an axis-aligned bounding box.
- `add_polygon_segmentation(label, slice_index, points)` places an arbitrary polygon, points given as $[[x_1, y_1], \ldots]$.

A.3.3. Reporting

Reporting tools produce the structured deliverable for tasks that require one. Some are terminal and end the episode when invoked; others are called multiple times per task, accumulating annotations until a terminal tool is invoked.

- `submit_birads_report(laterality, lesion_count, birads_category, enhancement_present, findings?, recommendation?)` submits a structured breast-MRI report. The optional `findings` array holds per-lesion fields (`location_quadrant`, `type` $\in$ {`mass`, `non_mass`, `focus`}, `size_mm`, `shape`, `margin`, `enhancement`) following the BI-RADS MRI lexicon.
- `submit_longitudinal_finding(finding_type, slice_index, location, description?)` logs one longitudinal finding, with `finding_type` $\in$ {`new_lesion`, `size_change`, `no_change`} and `location` in follow-up pixel coordinates. Unlike the other reporting tools, this one is *not* terminal and is called once per finding.
- `submit_longitudinal_complete(summary?)` terminates a longitudinal comparison task after all findings have been submitted.
- `submit_answer(answer)` submits a free-text answer for metadata-QA tasks.

B. Details of ABRA Benchmark

B.1. Dataset selection

ABRA draws its studies from three public collections, all accessed through TCIA (Clark et al., 2013). Together they span the three most common radiology modalities (CT, DX/CR, MR) and two annotation schemas carried by DICOM itself: SEG volumes for pixel-level contours and SR objects for structured reports. Table 6 summarises the resulting cohort after local filtering, and Table 7 gives the corresponding task-type contributions.

Table 6. Datasets used by ABRA after local filtering. *Studies* counts distinct `StudyInstanceUIDs`; *Series* counts individual DICOM series including derived objects (SEG for segmentation, SR for structured reports). Parenthesised numbers in the modality column give per-modality series counts.

Dataset	Domain	Patients	Studies	Series	Modalities (series)	Reference	License
LIDC-IDRI	Thoracic CT	10	20	142	CT (10), DX (8), CR (2), SEG (61), SR (61)	Armato et al. (2011)	CC BY 3.0
Duke Breast Cancer MRI	Bilateral breast MRI	50	50	278	MR (270), SEG (8)	Saha et al. (2021)	CC BY-NC 4.0
NLST New-Lesion LongCT	Longitudinal thoracic CT	121	246	246	CT (246)	Gong et al. (2025)	CC BY 4.0

Table 7. Contribution of each source dataset to the 655 ABRA tasks, with task-type breakdown. The tasks-per-study ratio (rightmost column) is a measure of annotation density: LIDC-IDRI's small study count is offset by the presence of multiple expert-consensus nodules per CT and by rich supplementary series (DX/CR radiographs, SEG, SR) on every study, which together support five different task types from only 20 studies. Duke and NLST tasks are drawn far less densely, reflecting single-modality content with at most one reporting target or one longitudinal finding per case.

Dataset	Studies	Task types sourced	Tasks	Tasks / study
LIDC-IDRI	20	`viewer_control`, `metadata_qa`, `annotation`, `oracle_annotation`, `vision_probe`	326	16.3
Duke Breast	50	`oracle_birads_report`, `birads_report`, `vision_probe`	135	2.7
NLST LongCT	246	`metadata_qa`, `longitudinal`, `vision_probe`	194	0.8
Total	**316**		**655**	

LIDC-IDRI. LIDC-IDRI is used for every task type that touches thoracic CT: the viewer-control and metadata-QA tiers, the annotation and oracle-annotation tasks, and a subset of the image-only vision probes. For each of 10 patients we retain a primary CT study together with its accompanying chest radiograph study, giving 20 DICOM studies and 142 series in total (10 CT volumes, 8 DX and 2 CR radiographs, 61 SEG objects, and 61 SR reports). Nodules in LIDC-IDRI are contoured independently by up to four thoracic radiologists per patient, and each reader's contour is released as a separate DICOM SEG object, which accounts for the SEG and SR series counts. To obtain a single reference volume per nodule we apply a volumetric 50% majority-vote consensus, following the convention adopted in the LIDC-IDRI community: per-reader three-dimensional masks are zero-padded over the union z-range of all readers who marked the same nodule, averaged, and thresholded at ≥ 0.5, so that a slice marked by only one of three readers is discarded rather than passing through as a single-reader record. After consensus, the retained subset contains 22 distinct nodules across 9 patients (one patient has no nodule that clears the consensus threshold). These consensus volumes serve as the reference for the IoU-based outcome scorer in annotation tasks. Although the retained LIDC-IDRI cohort is small in patient count, the combination of multi-reader consensus nodules, supplementary radiograph studies, and DICOM-native derived objects yields 326 tasks (16.3 per study), covering five of the eight task types in the benchmark.

Duke Breast Cancer MRI. The Duke cohort supplies bilateral breast MRI studies with clinical BI-RADS metadata, and is the basis for both BI-RADS reporting task types (`oracle_birads_report` and `birads_report`). We retain 50 patients (50 studies, 278 series) whose dynamic contrast-enhanced (DCE) acquisitions are complete. A typical study contains a pre-contrast axial sequence, four post-contrast DCE passes, and a contrast-enhanced T1-weighted series, giving the five to seven MR series visible to the agent at task time. DICOM SEG segmentations are available for a subset of lesions but are not required by the reporting task formulation. Structured BI-RADS attributes (laterality, lesion count, BI-RADS assessment category, enhancement status) are taken from the clinical spreadsheets released with the collection.

NLST New-Lesion LongCT. NLST-LongCT provides baseline and follow-up low-dose chest CT pairs with expert-annotated new-lesion coordinates, and is the sole source of `longitudinal` tasks. We ingest 121 participants and 246 CT studies, comprising 125 baseline/follow-up pairs as released by NLST. The released collection partitions into 90 pairs with a single new lesion, 14 with multiple new lesions (up to six), and 21 with no new lesion (released as longitudinal negatives by NLST but not used in ABRA, since the longitudinal task formulation requires a lesion to localise). After excluding one pair with a malformed annotation during task construction, 124 are retained for longitudinal task generation. The curated annotations record 129 new-lesion points across the lesion-bearing pairs. NLST also contributes the `metadata_qa` templates that query longitudinal DICOM tags (study-date interval and follow-up slice count) and a share of the modality and preprocessing vision probes.

B.2. Task generation templates

ABRA tasks are produced by deterministic generators. Each generator instantiates a fixed text template with fields drawn from the dataset and per-study metadata (patient ID, study UID, series UID, slice ranges, ground-truth labels). The resulting `task_description` string is written to the task YAML and later appended to the agent system prompt at runtime. Placeholders in Table 8 are written in the form {`field`} exactly as they appear in the generator source.

Agent system prompt. The runtime system prompt is assembled during benchmark runtime. The preamble is fixed; the study-context block is included whenever a `study_uid` is set on the task; the viewer-state block is included; and the task description is appended last. The viewer-state JSON contains a focused subset of the viewport state (`sliceIndex`, `totalImages`, `windowWidth`, `windowCenter`, `zoom`, `seriesInstanceUID`, `displaySetInstanceUIDs`).

```
You are a radiology AI agent operating inside a medical imaging viewer.
Use the available tools to complete the task described below. Be precise
and efficient -- use only the tools necessary to complete the task. All
coordinates are in pixel space.

Study context:
- StudyInstanceUID: {study_uid}
- SeriesInstanceUID (loaded): {initial_series_uid}

Current viewer state:
{viewer_state_json}

Task: {task_description}
```

Figure 4. Agent system prompt template. {`task_description`} is the per-task string from Table 8; {`viewer_state_json`} is a JSON dump of the current OHIF viewport state.

Per-task templates. Table 8 lists every `task_description` template used by the benchmark, grouped first by difficulty tier and then by task type. Near-duplicate probes that differ only in pixel-preprocessing pipeline (such as the *normal* and *noise* variants of a vision-probe classifier) share the same template and are collapsed into a single row. A literal `\n` denotes a newline in the rendered prompt.

Table 8. Task-description templates used by the ABRA generators, grouped by difficulty tier and task type. Placeholders {`field`} are instantiated at generation time from study, series, and annotation metadata.

Task ID prefix	Template
Easy · `viewer_control`	
`t1_slice`	`Navigate to slice {target} of the current CT series.`
`t1_wl_*`	`Set the window width to {preset.ww} and window center to {preset.wc} for a standard {preset.label} on this {patient_id} chest CT.`
`t1_slice_wl`	`Navigate to slice {target} and apply a bone window (window width 2500, window center 480) on this {patient_id} CT.`

Table 8 continued from previous page.

Task ID prefix	Template
t1_series	The current viewport shows a CT series from {patient_id}. First query the study metadata to discover available series, then select the {target.modality} series (``{target.description}'').
Easy · metadata_qa	
t2_slices	How many CT image slices are in the {patient_id} study? Query the series metadata and count the instances in the CT series. Answer with only the integer count.
t2_nseries	How many series are in the {patient_id} study (StudyInstanceUID: {study_uid})? Count all series regardless of modality. Answer with only the integer count.
t2_modalities	What distinct imaging modalities are present in the {patient_id} study? Query the series metadata and list all unique modality values, sorted alphabetically and separated by commas. Answer with only the comma-separated list (e.g., 'CT, SEG, SR'), no other text.
t2_date	What is the study date (StudyDate DICOM tag) for the {patient_id} study? Answer with only the 8-digit date in YYYYMMDD format (e.g., '20000101'), no other text.
t2_ct_uid	What is the SeriesInstanceUID of the CT series in the {patient_id} study? Query the series metadata and find the series with modality CT. Answer with only the SeriesInstanceUID string, no label or other text.
t4_interval	What is the time interval (in days) between the baseline study (StudyInstanceUID: {baseline.study_uid}) and the follow-up study (StudyInstanceUID: {followup.study_uid}) for participant {participant_id}? Query the study metadata for both studies and compute the difference in StudyDate values. Answer with only the integer number of days, no other text.
t4_slice_diff	How many more (or fewer) CT slices does the follow-up study have compared to the baseline for participant {participant_id}? Baseline StudyInstanceUID: {baseline.study_uid}. Follow-up StudyInstanceUID: {followup.study_uid}. Query series metadata for both studies, find the CT series in each, and report the difference (follow-up minus baseline). Use a positive number if follow-up has more slices, negative if fewer. Answer with only the signed integer (e.g., '-17' or '24'), no other text.
Easy · vision_probe	
vp_mod	What modality is this image?\nA) CT\nB) MRI\nC) DX\nD) N/A\n\nRespond by calling submit_answer with only the letter.
vp_pre	What windowing preset was applied to this image?\nA) Lung window\nB) Soft tissue window\nC) Default\nD) Breast MRI\nE) N/A\n\nRespond by calling submit_answer with only the letter.
Medium · annotation	
t3_nodule	Navigate to slice {slice_index} of the CT series and place a segmentation annotation on the pulmonary nodule (``{segment_label}'') in this {patient_id} chest CT. Apply a lung window (WW: 1500, WC: -600) for optimal visualization. Use a circle or polygon region to outline the nodule.
t3_find	Find and segment the nodule labeled ``{label}'' in this {patient_id} chest CT. The nodule is visible on slices {first_slice} through {last_slice} ({num_slices} slices). Navigate to each slice, apply a lung window, inspect the image, and place a segmentation annotation on every slice where this specific nodule is present.
Medium · oracle_annotation	

Table 8 continued from previous page.

Task ID prefix	Template
t3_oracle (single)	Use the external pathology detection model to identify and segment the pulmonary nodule in this {patient_id} chest CT. First query the model for an overview of findings in the CT series, then request the precise segmentation contour for the recommended slice. Navigate to that slice and place the annotation using the model's output.
t3_oracle (multi)	Use the external pathology detection model to identify and segment all pulmonary nodules in this {patient_id} chest CT. Query the model for an overview of all findings, then for each finding request the precise contour, navigate to the slice, and place an annotation. Annotate all findings.
t3_oracle (volumetric)	Use the external pathology detection model to segment the nodule ``{segment_label}'' across all its slices in this {patient_id} chest CT. Query the model for an overview, then for each slice in the nodule's range, request the precise contour, navigate to the slice, and place the annotation.
Medium · oracle_birads_report	
t3_oracle_birads	You are viewing a breast MRI study for patient {patient_id}. An external breast MRI CAD model is available via the query_birads_model tool. Query the model for the loaded series to obtain BI-RADS findings, then submit a structured BI-RADS report using submit_birads_report with the model's findings including laterality, lesion count, BI-RADS category, and whether enhancement is present.
Hard · longitudinal	
t4_lesion (single)	Compare baseline and follow-up chest CTs for participant {participant_id}. A new lesion has appeared on the follow-up scan that was not present on the baseline. Baseline StudyInstanceUID: {baseline.study_uid} (series: {baseline.series_uid}). Follow-up StudyInstanceUID: {followup.study_uid} (series: {followup.series_uid}). First examine the baseline CT to understand the normal anatomy, then switch to the follow-up CT and navigate through slices to find the new lesion. Use get_dicom_image with lung_window preprocessor to view the images. Once you locate the new lesion, submit its location using submit_longitudinal_finding.
t4_lesion (multi)	Compare baseline and follow-up chest CTs for participant {participant_id}. Multiple new lesions may have appeared on the follow-up scan. Baseline StudyInstanceUID: {baseline.study_uid} (series: {baseline.series_uid}). Follow-up StudyInstanceUID: {followup.study_uid} (series: {followup.series_uid}). Examine both studies using lung window settings, identify all new findings, and submit each one using submit_longitudinal_finding. When done, call submit_longitudinal_complete.
Hard · birads_report	
t4_birads	You are viewing a breast MRI study (StudyInstanceUID: {study_uid}). This study contains {n_series} MR series including dynamic contrast-enhanced (DCE) sequences.\n\nAvailable MR series:\n{series_list}\n\nNavigate through the available series to identify any enhancing lesions. Compare pre-contrast and post-contrast sequences to assess enhancement. Produce a structured BI-RADS report by calling submit_birads_report with your findings including laterality, lesion count, BI-RADS category, and whether enhancement is present.

Per-task tool availability. Each task type fixes the tool set passed to the agent for the duration of an episode. Table 9 lists the full mapping. Tools that are irrelevant to a task's expected trajectory (for example, a reporting action in a segmentation task) are included whenever the task type admits them, so that planning can be penalised for inappropriate tool selection. Vision-probe episodes expose only a single submission tool, reflecting their classification-only nature;

oracle variants substitute an oracle-query tool for the annotation primitives; and only longitudinal tasks expose the two-step finding/complete reporting pair.

Table 9. Tool availability by task type in ABRA. A ✓ indicates that the tool is included in the tool set passed to the agent when a task of that type is loaded. Columns are ordered by difficulty tier (Easy, Medium, Hard).

	viewer_control	metadata_qa	vision_probe	annotation	oracle_annotation	oracle_birads_report	longitudinal	birads_report
Navigation and display								
set_viewport_slice	✓	✓		✓	✓	✓	✓	✓
set_window_level	✓	✓		✓	✓	✓	✓	✓
set_zoom	✓	✓		✓	✓	✓	✓	✓
select_series	✓	✓		✓	✓	✓	✓	✓
Metadata queries								
get_viewport_state	✓	✓		✓	✓	✓	✓	✓
get_study_metadata		✓		✓	✓	✓	✓	✓
get_study_series		✓		✓	✓	✓	✓	✓
get_series_metadata		✓		✓	✓	✓	✓	✓
get_instance_metadata		✓		✓	✓	✓	✓	✓
Pixel and viewer snapshot								
get_dicom_image				✓			✓	✓
get_viewer_screenshot				✓			✓	✓
Oracle predictions								
query_pathology_model					✓			
query_birads_model						✓		
Segmentation								
add_circle_segmentation				✓			✓	✓
add_rectangle_segmentation				✓			✓	✓
add_polygon_segmentation				✓	✓		✓	✓
list_segmentations				✓	✓		✓	✓
Reporting								
submit_answer		✓	✓	✓	✓	✓	✓	✓
submit_birads_report						✓		✓
submit_longitudinal_finding							✓	
submit_longitudinal_complete							✓	

B.3. Initial State Setup Details

Before each episode the controller issues a `task_reset` call to the viewer, which clears all prior annotations and in-memory viewport state, loads the task's `study_uid` (together with any auxiliary studies such as baseline/follow-up pairs for longitudinal tasks), selects the initial series specified by `initial_series_uid`, and navigates to `initial_slice_index`. The controller then queries `get_viewport_state` and embeds the result in the system prompt so the agent starts from a concrete, deterministic snapshot of the loaded study, isolating evaluation from ordering effects or leakage across episodes.

B.4. Evaluation

Each episode is scored on three dimensions whose weighted sum defines the per-task score:

$$S \;=\; 0.20 \cdot P \;+\; 0.30 \cdot E \;+\; 0.50 \cdot O, \qquad S,\, P,\, E,\, O \in [0, 1].$$

All three dimensions are evaluated for every task type; only the *Outcome* scorer is task-type specific.

B.4.1. PLANNING

Planning compares the agent's tool-call sequence to the task's reference trajectory as an *unordered multiset*:

$$P \;=\; \max\Big(0,\;\; F_1(T_{\text{ref}},\, T_{\text{agent}}) \;-\; \min\big(0.30,\; 0.05 \cdot \max(0,\; |T_{\text{agent}}| - |T_{\text{ref}}|)\big)\Big)\,,$$

where F_1 is computed over tool-name counts (precision $= \mathrm{TP}/|T_{\text{agent}}|$, recall $= \mathrm{TP}/|T_{\text{ref}}|$, $\mathrm{TP} = \sum_t \min(\mathrm{ref}[t], \mathrm{act}[t])$). The subtracted term is a *redundancy penalty* of 0.05 for each tool call beyond the reference length, capped at 0.30. Ordering is intentionally ignored so that agents can interleave exploratory reads without being penalised on Planning.

Reference trajectories T_{ref} are constructed deterministically by each task generator from the minimum tool sequence needed to satisfy the task; they are not learned from agent rollouts. Table 10 lists the construction for every task subtype. Some trajectories have a length that depends on per-task parameters: N is the number of slices a nodule spans, K is the number of findings returned by the oracle overview, L is the number of ground-truth lesions in a longitudinal pair, and V is the number of MR series shown to the agent (capped at 4).

Table 10. Reference trajectories used by the planning scorer, by task subtype. Tool names are compared as an *unordered multiset*, so ordering inside a row is shown only for readability. Bracketed blocks with a superscript (for example $[\,\text{svs, gdi}\,]^5$) are repeated inline.

Task ID prefix	Reference trajectory	Length
Easy		
`t1_slice`	`set_viewport_slice`	1
`t1_wl_*`	`set_window_level`	1
`t1_slice_wl`	`set_viewport_slice, set_window_level`	2
`t1_series`	`get_study_series, select_series`	2
`t2_slices`, `t2_modalities`, `t2_ct_uid`	`get_study_series, submit_answer`	2
`t2_nseries`, `t2_date`	`get_study_metadata, submit_answer`	2
`t4_interval`	`get_study_metadata, get_study_metadata, submit_answer`	3
`t4_slice_diff`	`get_study_series, get_study_series, submit_answer`	3
`vp_mod`, `vp_pre`	`submit_answer`	1
Medium		
`t3_nodule`	`get_study_series, set_viewport_slice, set_window_level, get_dicom_image, add_circle_segmentation`	5
`t3_find`	`get_study_series, set_viewport_slice, set_window_level, get_dicom_image, add_circle_segmentation, [set_viewport_slice, get_dicom_image, add_circle_segmentation]`$^{N-1}$	$5 + 3(N-1)$
`t3_oracle` (single)	`get_study_series, query_pathology_model, query_pathology_model, set_viewport_slice, add_polygon_segmentation`	5
`t3_oracle` (multi)	`get_study_series, query_pathology_model, [query_pathology_model, set_viewport_slice, add_polygon_segmentation]`^{K}	$2 + 3K$
`t3_oracle` (volumetric)	`get_study_series, query_pathology_model, [query_pathology_model, set_viewport_slice, add_polygon_segmentation]`^{N}	$2 + 3N$
`t3_oracle_birads`	`get_study_series, query_birads_model, submit_birads_report`	3
Hard		
`t4_lesion` (single)	`get_study_series, select_series, set_window_level, [set_viewport_slice, get_dicom_image]`^{5}`, select_series, set_viewport_slice, set_window_level, [set_viewport_slice, get_dicom_image]`^{5}`, submit_longitudinal_finding, submit_longitudinal_complete`	28
`t4_lesion` (multi)	`get_study_series, select_series, set_window_level, [set_viewport_slice, get_dicom_image]`^{5}`, select_series, set_viewport_slice, set_window_level, [set_viewport_slice, get_dicom_image]`^{5}`, [submit_longitudinal_finding]`^{L}`, submit_longitudinal_complete`	$27 + L$
`t4_birads`	`get_study_series, [select_series, [set_viewport_slice, get_dicom_image]`^{3}`]`^{V}`, submit_birads_report`	$2 + 7V$

B.4.2. EXECUTION

Execution aggregates four facets of how well the chosen tools were run (weights in parentheses):

$$E \;=\; 0.40 \cdot A_{\text{tool}} \;+\; 0.20 \cdot Q_{\text{param}} \;+\; 0.25 \cdot E_{\text{turn}} \;+\; 0.15 \cdot R_{\text{err}}.$$

- A_{tool}: tool-call accuracy, the fraction of calls that returned success.
- Q_{param}: parameter quality, a task-aware rubric checking that arguments are well-typed and consistent with the loaded study (for example, a window/level pair plausible for the modality or a slice index inside $[0,\ \text{totalImages})$).
- $E_{\text{turn}} = \min\big(1,\ |T_{\text{ref}}|/\max(1,\ n_{\text{turns}})\big)$: turn efficiency relative to the reference length.
- R_{err}: error-recovery score, rewarding agents that retry or replan after a failed tool call rather than repeating it identically.

B.4.3. Outcome

The outcome scorer is selected per task type (Table 11). All outcome scores are clipped to $[0, 1]$.

Table 11. Task-type-specific outcome scorers used in ABRA.

Scorer	Task types	Signal
State diff	`viewer_control`	Per-field match against `expected_outcome` on the final viewport state; partial credit for multi-field tasks.
Exact match	`metadata_qa`, `vision_probe`	Case- and whitespace-normalised equality between the submitted answer and the reference string.
IoU (normalised)	`annotation`, `oracle_annotation`	Raw intersection-over-union against the reference mask, divided by the IoU of the best-fit primitive of the agent's chosen shape: an area-matched circle centred on the reference centroid for circles; the tighter of the axis-aligned and minimum rotated bounding box for rectangles; fixed at 1 for polygons. A separate hit rate at normalised IoU≥ 0.5 is also reported.
Point distance	`longitudinal` (single finding)	1 within a 20-pixel radius of the reference point, linear falloff to 0 at 40 px, hard zero if the submission names the wrong slice.
Multi-finding	`longitudinal` (multiple findings)	Detection rate minus 0.1 per false positive, clamped to $[0, 1]$; matching uses the same slice and distance criterion as above.
BI-RADS report	`birads_report`, `oracle_birads_report`	Weighted field comparison (Table 12).

BI-RADS field weights. The BI-RADS report scorer is a weighted average over five fields. The assessment category and lesion count allow partial credit on off-by-one errors (0.5 for $|\Delta| = 1$), reflecting the ordinal nature of the BI-RADS scale and the clinical tolerability of small count discrepancies.

Table 12. Scored fields in the BI-RADS report scorer. Unscored free-text fields (per-lesion shape, margin, enhancement, type, size; textual recommendation) are retained in the trace for qualitative analysis.

Field	Weight	Match rule
`laterality`	0.25	Exact match (left/right/bilateral).
`birads_category`	0.30	Exact, or 0.5 for $|\Delta| = 1$.
`lesion_count`	0.20	Exact, or 0.5 for $|\Delta| = 1$.
`enhancement_present`	0.15	Boolean match.
`lesion_quadrant`	0.10	Exact match; omitted from the denominator when the reference is absent.

C. Baseline methods details

C.1. Per-task-type score breakdown

Table 13 gives the per-task-type Planning, Execution, and Outcome scores underlying the tier averages in Table 4. `vision_probe` is the only Easy task whose Outcome stays below 1.0 on every model, since modality and preprocessor identification is itself a small visual classification rather than a tag lookup or a viewport command. On the Hard tier, achieved Outcome concentrates on `birads_report` (0.06–0.64 across the roster) and is uniformly below 0.03 on `longitudinal`.

Table 13. Per-model, per-task-type Planning, Execution, and Outcome scores across the eight ABRA task types, grouped by difficulty tier. n is the number of tasks of the type.

Task type	Model	n	Plan ↑	Exec ↑	Outcome ↑	Avg ↑
Easy						
`viewer_control`	Claude Sonnet 4.6	60	0.98	1.00	1.00	1.00
	GPT-5.4	60	0.87	0.98	1.00	0.97
	GPT-5.4-nano	60	0.94	0.98	1.00	0.98
	Gemini 3 Flash	60	0.11	0.77	0.95	0.73
	Gemini 3 Pro	60	0.02	0.78	0.98	0.73
	Qwen 3.5	60	0.93	0.98	1.00	0.98
	Gemma 4	60	0.99	1.00	1.00	1.00
	Ministral 3 (14B)	60	0.94	0.99	1.00	0.99
	Mistral Large 3	60	0.75	0.92	0.97	0.91
	Kimi K2.5	60	0.20	0.83	1.00	0.79
`metadata_qa`	Claude Sonnet 4.6	90	0.82	1.00	1.00	0.96
	GPT-5.4	90	0.64	1.00	0.87	0.86
	GPT-5.4-nano	90	0.90	1.00	0.98	0.97
	Gemini 3 Flash	90	0.51	0.96	0.98	0.88
	Gemini 3 Pro	90	0.59	0.99	0.82	0.83
	Qwen 3.5	90	0.71	1.00	1.00	0.94
	Gemma 4	90	0.69	1.00	0.78	0.83
	Ministral 3 (14B)	90	0.82	0.98	0.56	0.74
	Mistral Large 3	90	0.89	1.00	0.79	0.87
	Kimi K2.5	90	0.96	1.00	0.87	0.93
`vision_probe`	Claude Sonnet 4.6	99	0.99	0.99	0.65	0.82
	GPT-5.4	99	0.99	0.99	0.83	0.91
	GPT-5.4-nano	99	0.99	0.99	0.70	0.84
	Gemini 3 Flash	99	1.00	1.00	0.75	0.87
	Gemini 3 Pro	99	1.00	1.00	0.64	0.82
	Qwen 3.5	99	1.00	1.00	0.77	0.88
	Gemma 4	99	1.00	1.00	0.88	0.94
	Ministral 3 (14B)	99	1.00	1.00	0.61	0.80
	Mistral Large 3	99	1.00	1.00	0.43	0.72
	Kimi K2.5	99	0.83	0.83	0.48	0.66
Medium						
`annotation`	Claude Sonnet 4.6	151	0.73	0.99	0.02	0.45
	GPT-5.4	151	0.80	0.99	0.03	0.47
	GPT-5.4-nano	151	0.66	0.98	0.00	0.42
	Gemini 3 Flash	151	0.57	0.96	0.25	0.53
	Gemini 3 Pro	151	0.65	0.99	0.16	0.51
	Qwen 3.5	151	0.73	0.89	0.00	0.42
	Gemma 4	151	0.74	0.91	0.08	0.46
	Ministral 3 (14B)	151	0.74	0.99	0.00	0.45
	Mistral Large 3	151	0.77	0.98	0.00	0.45
	Kimi K2.5	151	0.70	0.99	0.02	0.45
`oracle_annotation`	Claude Sonnet 4.6	51	0.90	1.00	1.00	0.98
	GPT-5.4	51	0.90	1.00	0.98	0.97
	GPT-5.4-nano	51	0.80	1.00	0.96	0.94
	Gemini 3 Flash	51	0.53	0.88	0.90	0.82

continued on next page

Table 13 continued from previous page

Task type	Model	n	Plan ↑	Exec ↑	Outcome ↑	Avg ↑
	Gemini 3 Pro	51	0.53	0.94	0.69	0.73
	Qwen 3.5	51	0.82	1.00	0.95	0.94
	Gemma 4	51	0.72	0.95	0.88	0.87
	Ministral 3 (14B)	51	0.87	0.96	0.90	0.91
	Mistral Large 3	51	0.89	1.00	0.91	0.93
	Kimi K2.5	51	0.51	0.93	0.84	0.80
`oracle_birads_report`	Claude Sonnet 4.6	50	0.80	1.00	1.00	0.96
	GPT-5.4	50	0.80	1.00	1.00	0.96
	GPT-5.4-nano	50	0.80	1.00	1.00	0.96
	Gemini 3 Flash	50	0.67	1.00	1.00	0.93
	Gemini 3 Pro	50	0.61	1.00	0.58	0.71
	Qwen 3.5	50	0.80	1.00	1.00	0.96
	Gemma 4	50	0.80	1.00	1.00	0.96
	Ministral 3 (14B)	50	0.80	1.00	1.00	0.96
	Mistral Large 3	50	0.80	1.00	1.00	0.96
	Kimi K2.5	50	0.80	1.00	0.94	0.93
Hard						
`longitudinal`	Claude Sonnet 4.6	104	0.05	0.98	0.00	0.30
	GPT-5.4	104	0.20	0.98	0.00	0.34
	GPT-5.4-nano	104	0.61	0.97	0.00	0.41
	Gemini 3 Flash	104	0.42	0.85	0.00	0.34
	Gemini 3 Pro	104	0.26	0.94	0.00	0.33
	Qwen 3.5	104	0.45	0.99	0.00	0.39
	Gemma 4	104	0.45	0.98	0.00	0.38
	Ministral 3 (14B)	104	0.47	0.95	0.02	0.39
	Mistral Large 3	104	0.55	0.88	0.00	0.37
	Kimi K2.5	104	0.38	0.98	0.01	0.38
`birads_report`	Claude Sonnet 4.6	50	0.53	0.99	0.64	0.73
	GPT-5.4	50	0.53	0.87	0.32	0.53
	GPT-5.4-nano	50	0.45	0.98	0.12	0.45
	Gemini 3 Flash	50	0.27	0.90	0.18	0.41
	Gemini 3 Pro	50	0.48	1.00	0.50	0.65
	Qwen 3.5	50	0.59	1.00	0.35	0.59
	Gemma 4	50	0.25	0.82	0.06	0.33
	Ministral 3 (14B)	50	0.36	0.71	0.35	0.46
	Mistral Large 3	50	0.43	0.94	0.47	0.60
	Kimi K2.5	50	0.58	0.99	0.42	0.62

C.2. Segmentation task analysis

Table 14. Per-model annotation behaviour on LIDC single-slice and volumetric annotation variants. *Slice match %* is the fraction of tasks where at least one agent annotation lands on a ground-truth axial slice. *Centroid* distances are computed only on slice-matched annotations. *Volume ratio* is per-pair agent-drawn area divided by reference polygon area, summed across slices. *Mode radius* is the single circle radius used most often, with the share of all circle annotations using it. Reference lesion area: median 120 px^2 (single-slice) and 62 px^2 (volumetric).

Model	Variant	n	Slice match % ↑	Centroid (px) ↓		Vol. ratio ↓		Circle %	Mode radius (px, %)
				med	p90	med	p90		
Claude Sonnet 4.6	single-slice	129	100	116	200	4.0	29.2	100	12 (67)
	volumetric	22	100	89	203	6.2	33.1	100	12 (25)
GPT-5.4	single-slice	129	98.4	90	262	2.1	14.9	97.6	8 (51)
	volumetric	22	100	65	181	3.3	17.7	78.2	7 (22)
GPT-5.4-nano	single-slice	129	100	104	177	5.9	45.2	95.3	18 (69)
	volumetric	22	81.8	70	154	9.6	55.0	100	18 (82)
Gemini 3 Flash	single-slice	129	98.4	34	206	2.7	16.1	100	10 (45)
	volumetric	22	100	55	188	2.4	16.1	100	5 (26)
Gemini 3 Pro	single-slice	129	98.4	44	198	2.1	16.1	100	10 (46)
	volumetric	22	95.5	11	169	1.9	17.5	100	5 (23)
Qwen 3.5	single-slice	129	86.8	102	206	3.1	36.6	99.1	18 (21)
	volumetric	22	90.9	74	130	3.9	44.7	100	10 (34)
Gemma 4	single-slice	129	89.9	85	161	2.2	16.1	100	10 (65)
	volumetric	22	95.5	85	224	2.7	16.1	100	10 (68)
Ministral 3 (14B)	single-slice	129	97.7	135	187	1.1	8.0	31.7	8 (55)
	volumetric	22	86.4	110	194	5.1	33.1	96.9	10 (31)
Mistral Large 3	single-slice	129	100	124	184	4.2	45.2	39.5	12 (49)
	volumetric	22	100	118	194	2.6	16.1	98.9	8 (68)
Kimi K2.5	single-slice	129	99.2	63	208	3.8	25.1	100	12 (39)
	volumetric	22	100	72	188	2.9	16.4	100	8 (27)

C.3. Oracle annotation analysis

Table 15. Per-model behaviour on LIDC oracle annotation tasks. The oracle tool returns the reference polygon (or per-slice polygons for the volumetric variant), and the task requires the agent to deposit those polygons in the viewer. *Queried* is the fraction of tasks where the agent invoked the oracle tool. Each task is then assigned to one of three mutually exclusive outcome buckets: *abstain* (no annotation submitted), *complete* (oracle polygons submitted verbatim on the correct slices), or alter (the residual, defined as $100 - \text{abstain} - \text{complete}$). The four columns under *Alter* characterise the alter cases and may overlap rather than sum: *wrong slice* (single-slice only) records placement on a non-ground-truth axial slice; *points modified* records submissions whose polygon differs from the oracle's; *extra* records submissions of more annotations than the oracle returned; *partial* (volumetric only) records cases where at least one ground-truth slice received no annotation.

Model	Variant	n	Queried % ↑	Abstain % ↓	Complete % ↑	Alter (% of tasks) ↓			
						Wrong slice	Points modified	Extra	Partial
Claude Sonnet 4.6	single-slice	23	100	0	100	0	0	0	n/a
	volumetric	21	100	0	100	n/a	0	0	0
GPT-5.4	single-slice	23	100	0	100	0	0	0	n/a
	volumetric	21	100	0	90.5	n/a	0	0	9.5
GPT-5.4-nano	single-slice	23	100	0	82.6	0	17.4	0	n/a
	volumetric	21	100	0	47.6	n/a	23.8	0	38.1
Gemini 3 Flash	single-slice	23	100	0	100	0	0	0	n/a
	volumetric	21	100	0	100	n/a	0	0	0
Gemini 3 Pro	single-slice	23	100	13	87	0	0	0	n/a
	volumetric	21	100	52.4	47.6	n/a	0	0	52.4
Qwen 3.5	single-slice	23	100	0	100	0	0	0	n/a

continued on next page

Table 15 continued from previous page

Model	Variant	n	Queried % ↑	Abstain % ↓	Complete % ↑	Alter (% of tasks) ↓			
						Wrong slice	Points modified	Extra	Partial
	volumetric	21	100	0	95.2	n/a	4.8	0	0
Gemma 4	single-slice	23	100	0	56.5	0	43.5	0	n/a
	volumetric	21	100	4.8	66.7	n/a	28.6	0	9.5
Ministral 3 (14B)	single-slice	23	100	0	95.7	0	4.3	0	n/a
	volumetric	21	100	0	66.7	n/a	33.3	0	4.8
Mistral Large 3	single-slice	23	100	0	100	0	0	0	n/a
	volumetric	21	100	0	57.1	n/a	4.8	0	42.9
Kimi K2.5	single-slice	23	100	21.7	78.3	0	0	0	n/a
	volumetric	21	100	0	52.4	n/a	14.3	0	42.9

C.4. Oracle BI-RADS analysis

Table 16. Per-model behaviour on Duke oracle BI-RADS reporting tasks ($n = 50$). The oracle tool returns the reference report fields; the task requires the agent to submit those fields verbatim. *Queried* is the fraction of tasks where the agent invoked the oracle tool. Each task is then assigned to one of three mutually exclusive outcome buckets: *abstain* (no report submitted), *complete* (oracle fields submitted verbatim), or alter (the residual). The five columns under *Alter* record which specific field of the report was modified. Surprisingly, Gemini 3 Pro always queried the oracle yet did not submit a final report in 42% of tasks, yielding Outcome $= 0$ on those episodes.

Model	n	Queried % ↑	Abstain % ↓	Complete % ↑	Alter (% of tasks) ↓				
					Laterality	Lesion count	Category	Enhancement	Quadrant
Claude Sonnet 4.6	50	100	0	100	0	0	0	0	0
GPT-5.4	50	100	0	100	0	0	0	0	0
GPT-5.4-nano	50	100	0	100	0	0	0	0	0
Gemini 3 Flash	50	100	0	100	0	0	0	0	0
Gemini 3 Pro	50	100	42	58	0	0	0	0	0
Qwen 3.5	50	100	0	100	0	0	0	0	0
Gemma 4	50	100	0	98	0	0	0	0	2
Ministral 3 (14B)	50	100	0	100	0	0	0	0	0
Mistral Large 3	50	100	0	100	0	0	0	0	0
Kimi K2.5	50	100	0	38	0	0	0	0	62

C.5. Longitudinal task analysis

Table 17. Per-model localisation error on single new-lesion longitudinal tasks ($n = 90$, NLST baseline-and-follow-up CT pairs). The agent must identify the new lesion appearing on the follow-up study and submit a single in-plane point on the slice carrying the finding. *Submitted* is the fraction of tasks with any submission. *Correct slice* requires exact match between submitted and ground-truth slice indices. *Slice diff*, *3D mm*, and *Within 100 mm* are computed over the submitted subset only. *3D mm* is the Euclidean distance combining in-plane error (PixelSpacing) and axial error (slice spacing) from per-series DICOM headers. *Hit* is the strict scorer outcome (slice match and in-plane distance within 20 px). The multi-lesion variant ($n = 14$) is omitted; no model produced a matched finding in the runs evaluated.

Model	n	Submitted % ↑	Correct slice % ↑	Slice diff ↓		3D mm ↓		Within 100 mm % ↑	Hit % ↑
				med	p90	med	p90		
Claude Sonnet 4.6	90	92.2	1.1	35	92	134	215	20.5	0
GPT-5.4	90	100	0	44.5	128	171	284	16.7	0
GPT-5.4-nano	90	94.4	0	67	121	166	259	12.9	0
Gemini 3 Flash	90	86.7	0	42.5	98	166	267	16.7	0
Gemini 3 Pro	90	95.6	3.3	43	112	167	267	16.3	0
Qwen 3.5	90	91.1	1.1	31	93	146	232	19.5	0
Gemma 4	90	100	0	36	99	179	308	15.6	0
Ministral 3 (14B)	90	98.9	0	45	98	146	208	18.0	0
Mistral Large 3	90	87.8	0	41	92	146	208	19.0	0

continued on next page

Table 17 continued from previous page

Model	n	Submitted % ↑	Correct slice % ↑	Slice diff ↓		3D mm ↓		Within 100 mm % ↑	Hit % ↑
				med	p90	med	p90		
Kimi K2.5	90	98.9	2.2	35	107	152	247	18.0	0

D. Limitations and future work

Programmatic reference trajectories. The Planning component compares each agent trajectory against a reference path generated programmatically from the task template at construction time. We acknowledge that this is our best estimate of a reasonable workflow rather than a radiologist-validated ground truth: the templates encode the steps that the task author judged necessary to solve the case, but they have not been reviewed against how a board-certified reader would actually drive the viewer. The empirical results offer indirect support for the construction. Planning and Execution scores are uniformly high on the easy and medium tiers (Table 4), consistent with the reference path capturing a plausible solution route when the workflow is short and well-constrained. On the hard tier, Planning scores diverge across models and drop relative to the easier tiers, which is the regime in which the reference's coverage of valid alternative paths is weakest and where independent validation would be most informative. The natural next step is to collect real radiologist trajectories on the same tasks. We plan to extend the OHIF viewer to log the tool calls a human reader produces while completing each task, so that radiologist-derived references can replace, or be aggregated with, the programmatic ones per task. ABRA already provides the underlying components for this (a task definition shared between agent and human, a viewer that exposes the same tool surface to both, and scorers that consume tool-call traces); the remaining work is the data collection itself.

Modality and dataset scope. ABRA currently samples three TCIA datasets (LIDC-IDRI chest CT, Duke breast MRI, NLST longitudinal CT). The patient counts are modest, particularly for LIDC at 20 studies, and the modality spread excludes mammography, plain radiography, ultrasound, PET-CT, and nuclear medicine; the datasets also share a single US-clinical sourcing population. Two observations help put this scope in context. First, the small patient counts are partly offset by per-study annotation density: LIDC's multi-radiologist consensus contours and per-nodule attributes yield many independent tasks per study, so the task budget is not directly bottlenecked by patient count. Second, and more fundamentally, ABRA is designed to measure agentic capabilities through proxy tasks rather than to estimate clinical performance on population-level radiology. Its purpose is to expose capability gaps and provide a measurable signal that improvement can be tracked against; population-scale clinical accuracy is a different question, requiring a different data design, that we do not claim to answer. The three datasets already span volumetric detection (LIDC), structured cancer reporting (Duke), and longitudinal change (NLST), which we view as a reasonable diversity floor for the questions ABRA is built to ask. Expanding modality coverage is possible as future work on ABRA. Each new modality requires its own task generators and reference data, so this is genuine engineering work; on the architectural side, however, ABRA tasks can be constructed from any DICOM-based dataset with annotations.

Radiology information system and FHIR integration. Working radiologists routinely consume non-imaging context (prior reports, demographics, labs, pathology) when interpreting a study, and several natural extensions of ABRA require that context to be addressable. The current environment exposes the imaging archive (DICOM via Orthanc) and the viewer, but no structured electronic-health-record interface, so tasks that depend on correlating imaging findings with longitudinal clinical data are presently out of scope. Adding a FHIR endpoint to the controller's tool surface is a clean architectural extension: an embedded FHIR server can be added alongside Orthanc and exposed through additional tools without changing the agent loop or the scoring framework. One concrete direction is to integrate with the FHIR sandbox of MedAgentBench, so that radiology and EHR tasks share a common environment and the two benchmarks become composable. The bottleneck for this extension is data rather than architecture. Public datasets that pair DICOM imaging with structured longitudinal EHR data for the same patients are rare, and the partial pairings we are aware of require credentialed access through institutional data-use agreements. Even MedAgentBench, which is EHR-only with no imaging component, sourced its patient profiles from a single institution and relied on practising physicians to author its tasks, illustrating that the EHR substrate alone is non-trivial to assemble. ABRA's FHIR integration is therefore a natural future direction whose realisation is data-bound rather than design-bound.

Multi-agent orchestration. ABRA is presently a single-agent benchmark: one agent receives the task, drives the viewer and other tools through the controller's tool surface, and produces the final submission. The oracle variants of the annotation and BI-RADS reporting tasks partly address the question this single-agent design might otherwise leave open. By providing the agent with ground-truth upstream context (the lesion location for oracle annotation, the finding text for oracle reporting),

these tasks mirror the situation in which an upstream perception or retrieval stage has succeeded and only the downstream subtask is being measured. The capability gap from upstream to downstream stages is therefore visible in the current results. What this design does not capture is error compounding across stages, the dynamic in which a perception subagent's miscalibrated output is consumed by a reporting subagent and propagates downstream. The direction in which specialised subagents collaborate, each on a small subtask of the workflow, is genuinely interesting, but evaluating it properly is out of scope for this work. Doing so on ABRA would require the scoring suite to attribute Planning, Execution, and Outcome to per-subagent contributions rather than to a single trajectory, which is a non-trivial change; it also depends on a task-decomposition convention that the agent literature has not yet converged on. Multi-agent orchestration is therefore a future direction whose realisation depends on both an extension of ABRA's scoring framework and broader progress on how subtasks should be defined.

Task realism and clinical scope. ABRA's current task set covers annotation, structured BI-RADS reporting, longitudinal new-lesion localisation, viewer control, and metadata querying. These are tractable proxies for radiologist behaviour rather than full reproductions of clinical work. A radiologist reading a chest CT in practice does not simply draw a circle around a nodule and submit; they reconcile the finding against the clinical indication, prior imaging, and the patient's broader history, then synthesise a free-text impression that serves the referring clinician. The proxies we score are deliberately narrow because narrow tasks have well-defined ground truth and admit automatic evaluation, but they leave large parts of the radiology workflow outside the benchmark. Bluethgen et al. (2025) sketch several higher-level agentic workflows that radiology agents would need to handle, including consistency checking on chest radiograph reports, end-to-end lung cancer screening reporting (scenario identification, longitudinal nodule tracking, Lung-RADS categorisation, and report drafting), agent-assisted resident tutoring, multidisciplinary team preparation, and follow-up scheduling. Each of these requires data and infrastructure that ABRA does not currently expose. Several further directions are also untouched in our task set: free-text full-report synthesis with paragraph-level scoring against a reference report, radiation treatment planning workflows, and similar workflow-level tasks that integrate imaging interpretation with downstream clinical actions. We do not claim that strong scores on ABRA imply readiness for any of these higher-level applications. The benchmark is positioned as a measurement of the perception and orchestration substrate that such applications would build on; closing the realism gap between this substrate and the workflows above is the principal direction of further work, and depends on the same data and scoring extensions discussed in the FHIR and multi-agent paragraphs above.

E. Total token budget per model

Table 18. Per-model token consumption and wall-clock cost across the full benchmark. Tokens are summed across tasks in each difficulty bucket as reported by each provider's billing-side counters. *Runtime* is the sum of per-task durations, which is independent of how many tasks were run in parallel during the benchmark.

Model	**Tokens (M)**				**Runtime (h)**
	Easy	Medium	Hard	**Total**	
Claude Sonnet 4.6	0.5	5.8	115.2	**121.4**	14.0
GPT-5.4	0.5	4.4	29.7	**34.7**	4.2
GPT-5.4-nano	0.5	4.9	34.7	**40.2**	3.6
Gemini 3 Flash	1.3	8.3	18.4	**28.0**	2.9
Gemini 3 Pro	1.4	5.6	53.1	**60.2**	16.0
Qwen 3.5	1.0	7.8	29.3	**38.0**	17.9
Gemma 4	0.6	4.5	11.6	**16.6**	16.4
Ministral 3 (14B)	0.9	5.8	11.9	**18.5**	3.5
Mistral Large 3	0.8	6.8	14.6	**22.2**	6.5
Kimi K2.5	1.1	8.5	25.3	**34.8**	17.2

F. Per-task-type token usage and latency

Table 19 reports mean per-task token usage, turn count, and wall-clock duration broken down by model and task type.

Table 19. Per-model, per-task-type token usage and latency, averaged over all tasks of the type. *Input* and *Output* are mean tokens per task. *Turns* is the mean number of agent turns per episode. *Dur.* is the mean wall-clock duration per task in seconds. Token counts are reported with a k suffix for $\geq 10^3$ and a M suffix for $\geq 10^6$.

Model	Task type	Input	Output	Turns	Dur. (s)
Claude Sonnet 4.6	viewer_control	3.0k	312	2.0	8.2
	metadata_qa	1.6k	259	2.1	6.4
	vision_probe	1.1k	101	1.0	10.2
	annotation	7	1.3k	5.4	29.1
	oracle_annotation	10	3.1k	7.7	41.3
	oracle_birads_report	4	372	2.0	7.3
	longitudinal	28	15.5k	25.6	327.5
	birads_report	10	3.7k	8.5	150.2
GPT-5.4	viewer_control	2.3k	52	2.6	3.4
	metadata_qa	2.9k	104	2.0	3.1
	vision_probe	1.0k	17	1.0	2.2
	annotation	13.4k	351	4.6	11.5
	oracle_annotation	17.1k	1.8k	5.3	23.5
	oracle_birads_report	3.1k	119	2.0	3.4
	longitudinal	146k	3.8k	9.4	93.0
	birads_report	36.9k	1.9k	4.3	35.7
GPT-5.4-nano	viewer_control	1.8k	73	2.2	4.0
	metadata_qa	3.1k	106	2.0	4.4
	vision_probe	1.0k	17	1.0	2.5
	annotation	13.9k	427	4.6	10.5
	oracle_annotation	22.9k	1.5k	6.7	16.6
	oracle_birads_report	3.1k	124	2.0	2.6
	longitudinal	174k	1.6k	20.8	75.2
	birads_report	24.9k	824	5.5	36.8
Gemini 3 Flash	viewer_control	11.4k	213	7.0	11.1
	metadata_qa	4.9k	238	2.7	4.9
	vision_probe	1.5k	16	1.0	4.5
	annotation	17.9k	499	4.9	13.5
	oracle_annotation	59.6k	2.9k	8.5	25.9
	oracle_birads_report	4.8k	232	2.9	4.8
	longitudinal	103k	1.7k	9.9	41.3
	birads_report	31.4k	822	5.2	20.3
Gemini 3 Pro	viewer_control	12.3k	977	7.8	28.8
	metadata_qa	3.2k	839	2.2	11.7
	vision_probe	1.5k	1.4k	1.0	16.0
	annotation	16.1k	12.3k	4.5	103.1
	oracle_annotation	13.4k	3.9k	4.9	62.4
	oracle_birads_report	4.1k	769	2.4	16.6
	longitudinal	273k	29.4k	8.7	270.4
	birads_report	114k	9.4k	6.3	115.7
Qwen 3.5	viewer_control	3.4k	478	2.4	14.2
	metadata_qa	5.0k	653	2.0	11.0
	vision_probe	1.2k	1.3k	1.0	32.7
	annotation	29.8k	2.0k	5.8	56.6
	oracle_annotation	57.7k	4.4k	7.9	114.8
	oracle_birads_report	5.4k	567	2.0	10.3
	longitudinal	211k	8.8k	15.1	351.7
	birads_report	122k	6.0k	13.4	177.5
Gemma 4	viewer_control	2.0k	64	2.0	8.3
	metadata_qa	3.9k	141	2.0	9.0
	vision_probe	731	15	1.0	8.0
	annotation	17.1k	450	4.1	98.7
	oracle_annotation	33.6k	2.3k	7.5	162.0
	oracle_birads_report	4.1k	141	2.0	23.6
	longitudinal	94.9k	1.6k	11.3	283.7

continued on next page

Table 19 continued from previous page

Model	Task type	Input	Output	Turns	Dur. (s)
	birads_report	30.2k	622	5.2	85.1
Ministral 3 (14B)	viewer_control	2.4k	187	2.3	3.9
	metadata_qa	6.5k	174	2.5	4.1
	vision_probe	1.1k	15	1.0	5.4
	annotation	23.4k	561	5.3	17.2
	oracle_annotation	34.3k	2.9k	6.1	31.3
	oracle_birads_report	4.5k	141	2.0	2.9
	longitudinal	76.8k	1.6k	11.7	45.7
	birads_report	71.9k	2.4k	9.0	45.3
Mistral Large 3	viewer_control	4.5k	121	3.4	13.1
	metadata_qa	4.3k	132	2.0	7.8
	vision_probe	1.1k	11	1.0	5.7
	annotation	24.5k	551	5.5	24.5
	oracle_annotation	51.3k	2.4k	7.9	44.2
	oracle_birads_report	4.5k	169	2.0	6.6
	longitudinal	117k	1.9k	15.1	125.5
	birads_report	43.0k	1.7k	7.7	42.9
Kimi K2.5	viewer_control	8.7k	643	6.8	21.5
	metadata_qa	3.2k	451	2.0	10.5
	vision_probe	1.2k	740	1.0	17.3
	annotation	19.3k	1.2k	5.8	45.3
	oracle_annotation	101k	3.5k	12.8	58.7
	oracle_birads_report	2.8k	373	2.0	7.1
	longitudinal	190k	4.3k	11.8	379.8
	birads_report	98.8k	3.5k	11.6	167.3