HealthAgentBench: A Unified Benchmark Suite of Realistic Agentic Healthcare Environments for Challenging Frontier AI Agents

Qianchu Liu*, Sheng Zhang*, Guanghui Qin*,
Jeya Maria Jose Valanarasu, Maximilian Rokuss, Mingyu Lu, Timothy Ossowski,
Juan Manuel Zambrano Chaves, Cliff Wong, Peniel Argaw, Yashna Hasija, Mu Wei, Wen-wai Yim,
Qin Liu, Zilin Jing, Jason Entenmann, Naoto Usuyama, Tristan Naumann, Hoifung Poon

Microsoft Research

microsoft.github.io/HealthAgentBench/

Abstract

As AI agents become increasingly capable of complex, long-horizon reasoning, rigorous evaluation is essential for measuring progress toward real-world healthcare applications. However, existing healthcare benchmarks are often saturated, static, or narrow in clinical scope, limiting their ability to distinguish frontier systems or track progress. We introduce **HealthAgentBench**, a suite of 54 agentic healthcare tasks across 7 categories each with its unique environment. The benchmark suite spans diverse workflows throughout the patient journey and a broad range of modalities. Each task is designed to replicate an end-to-end clinical workflow: given minimal instructions, an agent must explore raw healthcare data, operate within a complex environment, and execute multi-step solutions that go beyond naive prompting, such as querying large clinical databases or interpreting gigapixel pathology images. All tasks are scored using binary success/failure criteria against expert-derived labels or human performance. A final task success rate is reported to provide a single, interpretable metric for HealthAgentBench overall performance for each agent. Evaluating frontier agents on HealthAgentBench, we find that overall task success rate remains low, underscoring the difficulty of the suite. The strongest and the most cost effective agent, Codex GPT-5.5, achieves only approximately 42% success rate. Beyond aggregate performance, HealthAgentBench reveals nuanced strengths and weaknesses across task categories. Frontier agents show promise in automatically developing research modeling pipelines over EHR data, but medical imaging remains especially challenging, particularly for Claude Code models, while Codex GPT-5.5 shows emerging capability. Tasks that combine large search spaces with compositional reasoning requirements remain difficult for all current agents. Together, these results suggest that HealthAgentBench provides a challenging and realistic benchmark with substantial room for future progress. We release our benchmark at `https://github.com/microsoft/HealthAgentBench`.

*Equal contribution.

1 Introduction

The field of AI research is undergoing a transition from large language models (LLMs) as isolated task solvers to agentic systems that can execute actions within complex environments. Traditional LLM benchmarks were largely designed around static inputs: a model receives a fixed prompt, often containing a short text context, and produces a single answer. For example, tasks such as MedQA [15] evaluate whether an LLM can answer a question given a concise clinical vignette. Nowadays, LLM agents equipped with tools, harnesses, and execution environments can go beyond fixed-context prompting: they can search, inspect, query, compute, and iteratively act on data distributed across files, databases, images, and software systems. As a result, developing agentic evaluation environments has become essential for accurately measuring the capabilities of these frontier systems.

This shift is especially important for healthcare, where many high-value applications require end-to-end reasoning over complex, multimodal, and large-scale data. Facilitating real clinical workflows often involves processing information that cannot be fully represented in a short prompt, such as 3D CT volumes, gigapixel pathology slides, longitudinal EHR databases, trial protocols, and heterogeneous clinical documents. These settings have historically been beyond the reach of conventional prompting approaches, either because the data exceeds the model context window or because solving the task requires multi-step interaction with tools and the environments. Agentic systems create the possibility of addressing these workflows more realistically, by allowing models to explore the environment, operate over raw data, and design and execute task-specific strategies.

It is therefore crucial to move beyond static evaluation and develop realistic, holistic benchmark suites that assess the emerging capabilities of agents on end-to-end healthcare tasks. However, the current healthcare benchmark landscape remains fragmented and insufficient for evaluating the full capabilities of AI agents. Traditional benchmarks primarily measure isolated reasoning or question-answering ability, while recent agent-oriented efforts [7, 14, 21, 5] remain limited in scope, often focusing on specific tasks, modalities, or predefined workflows. This lack of broad, agent-native evaluation makes it difficult to draw reliable conclusions about the capabilities of frontier systems in healthcare, as real-world clinical workflows rarely rely on a single task, modality, or data source (eg. clinicians routinely integrate information from diverse inputs—such as pathology slides, CT scans, radiology reports etc. to arrive at diagnoses and treatment decisions). A comprehensive, realistic, and challenging benchmark suite is therefore needed to provide a common ground for evaluating and comparing agentic AI systems in healthcare, enabling researchers to identify strengths and areas for improvement and to track progress in this rapidly evolving field.

In this study, we introduce HealthAgentBench, a unified suite of agentic healthcare environments for evaluating AI agents. HealthAgentBench consists of 54 tasks across 7 categories each with its unique environment, including X-ray Report Correction, Pathology Tumor Area Selection, CT Abnormality Classification, Clinical Trial Matching, EHR Data Quality Auditing, EHR Event Modelling, and EHR Format Conversion. These tasks span diverse clinical workflows throughout the patient journey, including data management, diagnosis, research, and treatment planning. They also cover a broad range of modalities, including 2D radiographs, 3D CT volumes, whole-slide pathology images, free text, and structured clinical data. As detailed in Section 3, we provide a principled workflow to select and create each task in HealthAgentBench to ensure a unified benchmark that is realistic, challenging, verifiable and diverse. For each task, we provide a terminal-based environment following Merrill et al. [24] as shown in Figure 1. Within each environment, agents are given real clinical artefacts, including patient data and supporting resources, along with minimal task instructions. This setup allows agents to freely explore the environment, formulate strategies, and propose solutions. The X-ray Report Correction task shown in Figure 1 hands the agent an

environment containing a patient's longitudinal chest X-ray history and a corrupted report, and the agent can view, zoom into, and crop the radiographs and cross-reference the prior studies before writing a corrected report. In this way, HealthAgentBench evaluates agents' end-to-end autonomous capabilities, including planning, tool use, environment exploration, and task execution. For another example, when presented with a CT volume, an agent may choose to inspect slices sequentially, install image-processing tools, pre-filter relevant regions, or crop candidate lesions before making a final prediction. Finally, we provide a unified evaluation module to assess agent output by defining success/failure criteria for each task against human performance or expert labels, enabling consistent comparison of agents across the entire suite. Manual review is conducted to ensure that the success criteria are reachable.

We use HealthAgentBench to conduct an empirical study of frontier LLM agents. We find that the tasks are challenging even for state-of-the-art systems, with the strongest agent, Codex GPT-5.5, achieving only around 42% task success rate. Beyond this aggregate result, the suite enables nuanced analyses of agent strengths and weaknesses across task categories, clinical workflows, and data modalities, highlighting areas where current agents show promise as well as where substantial progress is still needed.

Contributions

In summary, our contributions are as follows:

1. **A unified suite of agentic healthcare environments.** We introduce HealthAgentBench, a benchmark suite for evaluating AI agents on realistic, patient-grounded tasks spanning diverse clinical workflows and data modalities, including medical imaging, free text, and structured EHR data. We also setup a principled workflow for sourcing and selecting tasks for the benchmark that can be extended for future editions.

2. **A challenging benchmark far from saturation.** HealthAgentBench is designed to measure progress in frontier healthcare agents over time. Current frontier agents achieve overall low task success rate, with the strongest model, Codex GPT-5.5, reaching only around 42%, leaving substantial room for future improvement.

3. **Nuanced analyses of frontier-agent capabilities.** Our benchmark reveals two major bottlenecks for current frontier agents: (i) medical imaging tasks (ie. CT, xray, pathology slides) (ii) tasks requiring large search spaces and complex compositional reasoning. We further observe clear model-family differences, with Codex GPT models, especially GPT-5.5, being generally more cost-effective and consistently outperforming Claude Code models on medical imaging tasks.

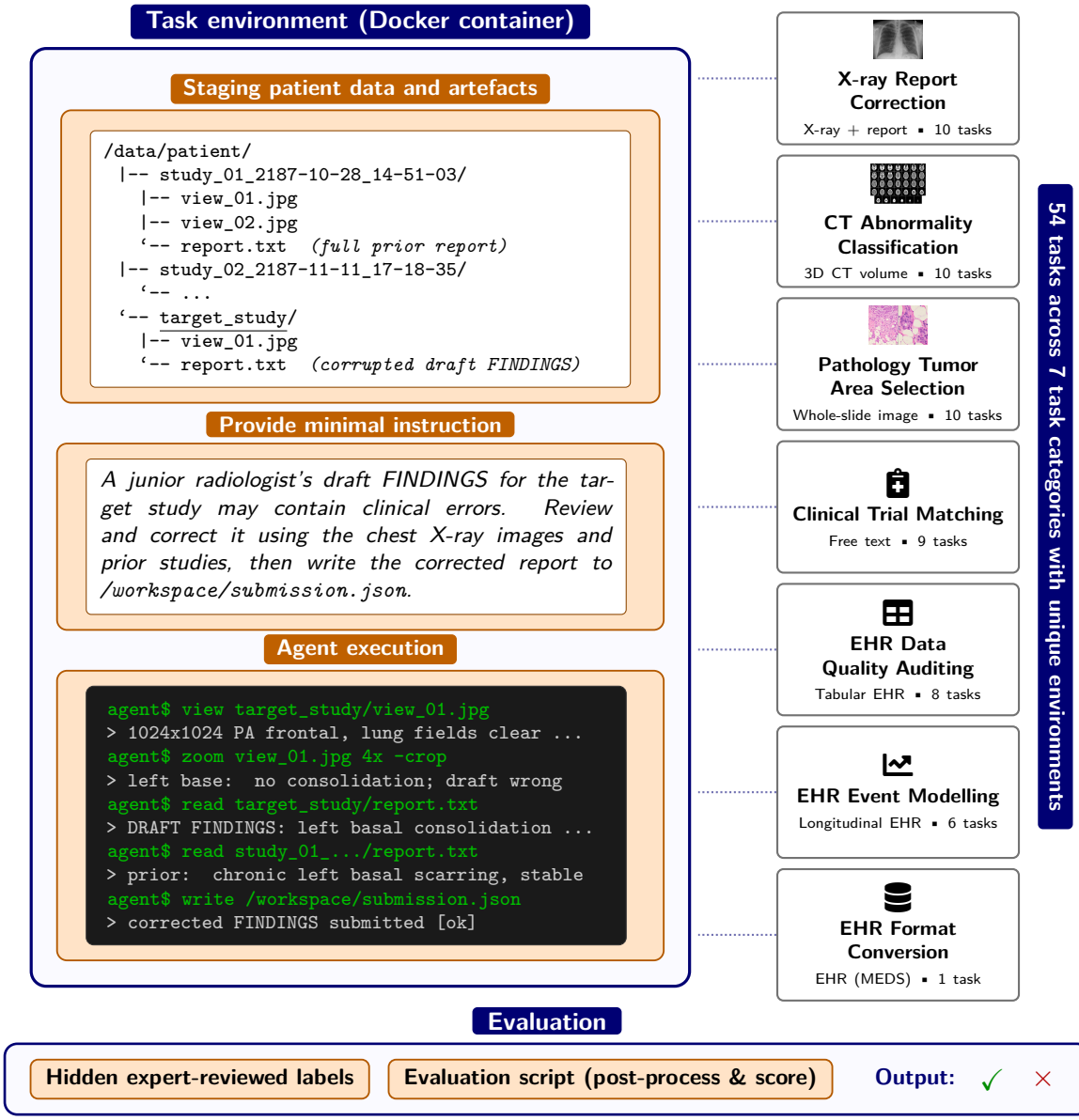


Figure 1: **HealthAgentBench is a unified evaluation framework for agentic healthcare AI.** Each benchmark task is packaged as a Docker container environment with three stages: (1) the container sets up the environment by staging patient data and necessary artefacts under `/data/` (here, the X-ray Report Correction task consists of one patient's longitudinal chest-X-ray history where the target study's report carries a corrupted draft FINDINGS section the agent must correct); (2) the agent is handed a natural-language instruction; (3) the agent acts in a terminal, issuing tool calls (it has freedom to decide whether to install packages or run commands to view, crop, and zoom into the images or consult the patient's prior studies) until it writes the corrected report to `/workspace/submission.json`. Evaluation runs outside of the container against hidden expert-reviewed groundtruth and a binary success criterion. The same containerized framework instantiates all 7 task categories (right), each in a unique environment, with the input modality and number of tasks shown for every category.

2 Related Work

Healthcare has a fast-growing body of agent-oriented benchmarks, but none yet offers a unified, executable, and multi-modal evaluation spanning the full patient journey. We group recent efforts into three complementary lines of work and discuss how HealthAgentBench relates to each.

A first line of work studies how agentic strategies can improve medical question answering. Two representative examples are MedAgentBoard [39], which offers a careful comparison of LLM multi-agent collaboration against single-LLM prompting and strong conventional methods across medical task families, and ClinicalAgent Bench [20], which considers a broad task suite spanning five clinical capability dimensions together with a rich clinical toolbox. These benchmarks are primarily organized around question-and-answer formats, in which each instance pairs a query with pre-supplied context and the agent's tools or collaborators act as part of the solver. HealthAgentBench is different in focus: rather than answering a self-contained question, the agent must explore and act within a live environment whose state it changes over many steps.

A second line of work develops genuinely interactive evaluations centered on clinical dialogue. HealthBench [6] and HealthBench Professional [13] provide carefully constructed, physician-authored rubrics for multi-turn patient and clinician conversations; MAI-DxO [26] models diagnosis as a cost-aware sequential process over challenging NEJM cases; and AgentClinic [31] turns medical QA and EHR cases into a simulated clinic with interacting doctor, patient, and measurement agents. In a similar spirit, simulated-hospital environments such as AI Hospital [8], MedAgentSim [1], and CP-Env [38] let models role-play clinical staff across multi-step encounters and branching care pathways. This body of work captures clinical reasoning and communication especially well. Its focus is conversational and simulated-patient interaction, whereas HealthAgentBench asks agents to operate directly on raw, heterogeneous patient data.

A third line of work, closest to ours, places agents in executable environments over real data, with each benchmark concentrating on a particular clinical setting. MedAgentBench [14] and the more recent PhysicianBench [22] situate agents within a FHIR-based EHR to retrieve records and place orders, with PhysicianBench extending the paradigm to long-horizon, execution-verified workflows; both center on structured EHR data. MedAgentGym [35] provides a scalable agentic environment for code-centric reasoning over biomedical data-science tasks, emphasizing programmatic analysis of structured records. AutoMedBench [21] and CamylaBench [9] share our emphasis on end-to-end pipelines over raw data, but they prescribe staged workflows rather than allowing agents to explore autonomously, and they focus primarily on medical imaging. HealthAdminBench [7], meanwhile, studies computer-use agents on healthcare administrative workflows. HealthAgentBench complements these efforts by unifying their strengths into a single, broad, executable benchmark in which agents autonomously interact with real, heterogeneous patient data. The suite spans five data modalities (2-D radiographs, 3-D CT volumes, pathology whole-slide images, free-text clinical documents, and structured EHR records) and four major stages of the clinical workflow: diagnosis, data management, research, and treatment planning. By adopting terminal-based environments with minimal instructions, HealthAgentBench builds on the design philosophy of general-domain agent benchmarks such as Terminal-Bench [24], OSWorld [34], GAIA [25], and WebArena [37], while adapting it to the multimodal data, complex workflows, and specialized tools characteristic of healthcare. Figure 2 positions HealthAgentBench relative to existing healthcare agent benchmarks along two dimensions: interaction realism and breadth of coverage. HealthAgentBench uniquely combines realistic agent interaction with broad coverage of clinical modalities and workflow stages, filling a key gap in the current evaluation landscape.

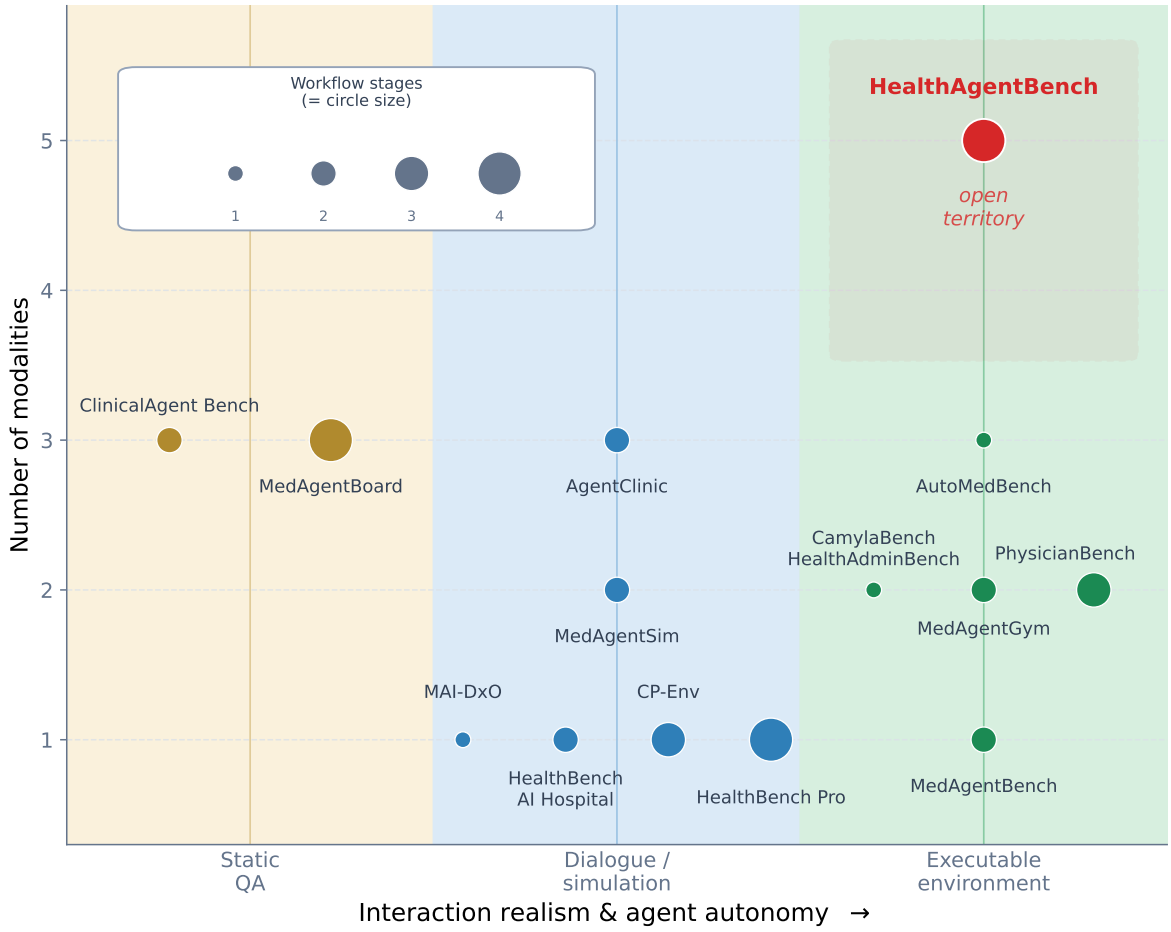


Figure 2: **Positioning of healthcare agent benchmarks.** An encoding of the per-benchmark comparison in Table 1 (Appendix A). The horizontal axis is *interaction realism and agent autonomy*, with the three interaction categories shown as distinctly coloured columns (static question answering, conversational/simulated interaction, and acting in an executable environment); benchmarks are pinned to their column's centre, so horizontal position within a column carries no meaning. The vertical axis is the *number of modalities* a benchmark covers. The *number of clinical workflow stages* is encoded as the size of each marker–a larger circle means more stages (see the key, top left). HealthAgentBench (red) is the sole occupant of the top-right "open territory" covering both high interaction realism and broad coverage of modalities and workflow stages.

3 Task Creation

We propose a principled workflow for selecting and designing tasks in HealthAgentBench to create 54 tasks across 7 task categories including X-ray Report Correction, Pathology Tumor Area Selection, CT Abnormality Classification, Clinical Trial Matching, EHR Data Quality Auditing, EHR Event Modelling and EHR Format Conversion. Documentation for tasks in each task category is presented in Appendix E. Figure 3 summarizes the selection criteria, construction sources, design principles, and post-creation checks that every task passes through. The workflow proceeds through four sequential stages: a candidate is screened against the *selection criteria*, then constructed from a vetted *source*, built under a fixed set of *design principles*, and finally subjected to *post-creation checks* before it joins the suite.

3.1 Task selection criteria.

To be included in HealthAgentBench, a candidate task must satisfy the following criteria

Agentic Workflow We deliberately choose tasks that require an *agentic* workflow that can involve planning, tool use, multi-step reasoning, or environment interaction that naive single-shot LLM prompting cannot complete. For example, the LLM cannot pass the ct volume, or pathology slide as a whole into the prompt.

Realistic Clinical Workflow The tasks cover different stages of the clinical workflow including data management (EHR Format Conversion, EHR Data Quality Auditing), diagnostics (X-ray Report Correction, Pathology Tumor Area Selection, CT Abnormality Classification, Clinical Trial Matching), event modelling (EHR Event Modelling) and treatment planning (Clinical Trial Matching). All tasks contain real clinical data including patient data or clinical documents such as trial information.

Wide coverage of modalities and environments Our task suite is designed to span diverse healthcare settings and environments. Figure 7 summarizes the coverage of HealthAgentBench across multiple dimensions, including input modality, clinical workflow, output format, patient and temporal scope, and data access regime. HealthAgentBench spans five input modalities—2-D radiographs, 3-D chest CT, pathology whole-slide images, free-text clinical documents, and structured EHR data. Its tasks also cover diverse agent output formats, including prose reports, ranked list, classifications, and engineered solutions such as ETL (Extract, Transform and Load) pipelines or flagged data-quality errors. The suite also varies patient scope, from single-record reasoning to cohort-scale analysis, and includes longitudinal tasks that require reasoning over time-ordered events.

Verifiable with low chance success rate We prioritize tasks with objectively verifiable success criteria while ensuring low chance success rate. In particular, we exclude single binary classification tasks with balanced labels, where random guessing achieves a success rate of 50%. Multi-label binary classification tasks are retained because the agent must make multiple independent predictions, making the probability of correctly guessing every label exponentially smaller (e.g., in CT Abnormality Classification, the agent must correctly identify the presence or absence of every abnormality in a CT volume to pass the task). Consequently, the tasks in HealthAgentBench have a low random-guess success rate—typically below 10%—ensuring that performance meaningfully reflects an agent's capabilities rather than chance.

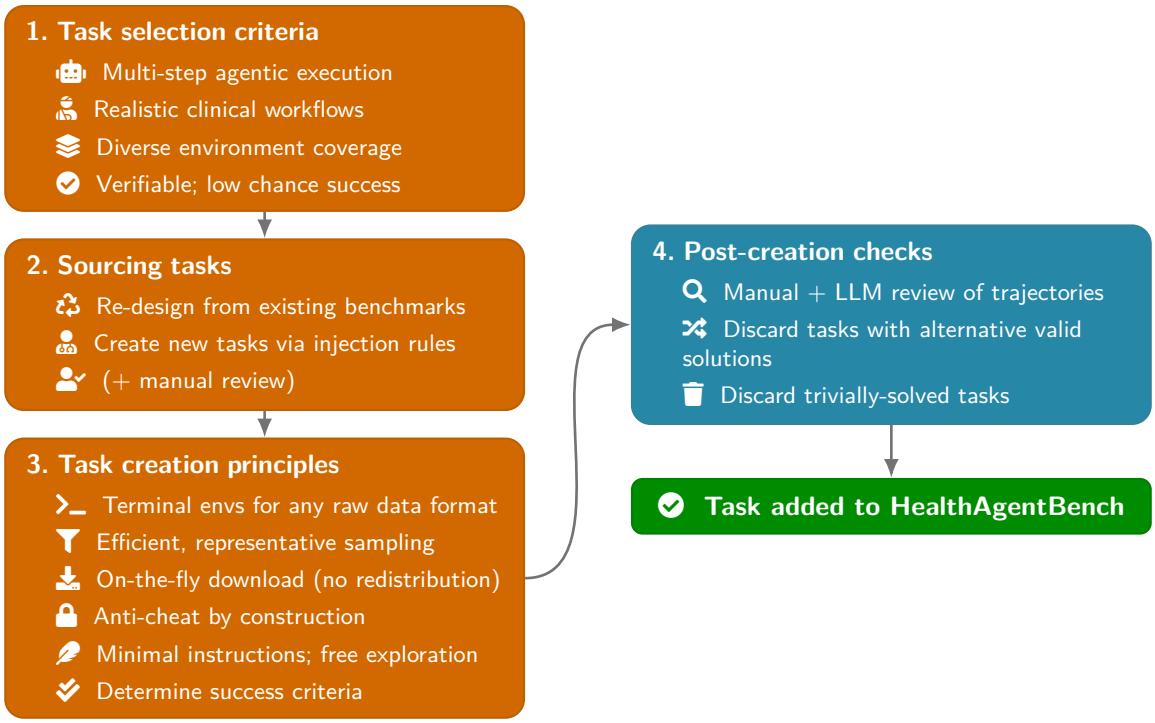


Figure 3: **The HealthAgentBench task-creation workflow.** Each candidate task progresses through four stages: (1) satisfying the selection criteria, (2) construction from either an existing benchmark or curated raw patient data, (3) application of standardized task design principles, and (4) final checks to eliminate opportunities for cheating, alternative valid solutions, and trivial tasks before inclusion in the benchmark.

3.2 Sourcing tasks.

Candidate tasks are constructed from two sources. First, we leverage existing benchmarks with expert labels or reports and convert them into terminal-based agent environments. We redesign the agent-visible interface so that agents must solve the tasks end-to-end through autonomous exploration, without the human scaffolding or task-specific engineering used in the original benchmark evaluations.For example, EHR Event Modelling is adapted from EHRSHOT [33]. Whereas EHRSHOT was originally designed for human researchers to develop predictive models, our version requires the agent to perform the entire machine learning workflow from exploring the patient database and selecting a modeling strategy to training a model and generating predictions on the test set, thereby transforming it into a substantially more challenging agentic evaluation. We also create new tasks by conducting rule-based augmentations or pipeline wrapping on patient data. Second, we create new tasks by augmenting curated patient datasets or wrapping existing clinical pipelines into executable environments. For example, EHR Data Quality Auditing is built by injecting realistic data quality issues into the MIMIC-IV dataset using rule-based perturbations, such as impossible physiological values inconsistent clinical records and demographic conflicts. The agent must inspect the underlying patient data, identify the injected errors, and report the corresponding data quality issues.

3.3 Task design principles.

Every task is built around five design principles as described below:

(i) Versatile terminal environment. We leverage the harbor framework [12] to package each task as a terminal environment that accommodates the heterogeneous raw patient-data formats the suite spans. Terminal environments provide a flexible interface through which agents can inspect heterogeneous data formats, invoke domain-specific software, query databases, and compose complex workflows over large-scale resources. They also closely match the execution environment of today's frontier coding agents, which are primarily designed and optimized to operate through terminal interfaces. This makes terminal environments a natural substrate for evaluating autonomous agent capabilities on realistic clinical tasks.[1].

(ii) Efficient, representative sampling. Rather than shipping hundreds of similar tasks, we sample a small set (typically 5–15 samples per task category) that are representative (eg. covering different disease types, difficulty levels) so a sweep of all tasks in this benchmark produces an informative signal in reasonable time.

(iii) On-the-fly data download. We do not redistribute data and labels, and provide a one-click run script to fetch data from the original sources at runtime after the user provides credentials.

(iv) Anti-cheat by construction. Gold labels and verifier code are mounted only into the verifier step, never into the agent container. Agent-visible identifiers are made opaque (e.g. `case_NN`, `study_NN_<timestamp>`) and corpus names are scrubbed from filenames so an agent cannot map back to the upstream dataset and look up answers. We also disable web browsing capabilities from agents to block the agent browsing internet for gold labels.

(v) Minimalist instructions. The agent-facing prompt states the goal in a few sentences but leaves the strategy (which files to read, in what order, which tools to call) entirely to the agent rather than predefining workflows. This way, we can test the innate planning and strategy formulation capabilities of the agents alongside their execution abilities.

(vi) Determine task-specific success criteria Each task is evaluated using a binary success/failure criterion, with task-specific performance metrics reported alongside. The success criterion is determined for each task category separately; Table 3 in Appendix B summarizes these criteria, and the per-task *Scoring* paragraphs in Section B give the full details. These criteria fall into a few broad types: *exact or complete correctness* (every CT abnormality label correct, or all ETL verifier checks passing), *exhaustive recovery* of the targets (full recall on the data-quality and trial-matching tasks), *error-free output* (zero clinically-significant errors for report correction), and *reaching a performance threshold*—either an absolute metric cutoff (tumor tile-F1 ≥ 0.90) or matching a human-engineered baseline (AUROC at or above the human engineered count+GBM model). In every case the threshold is set so that success reflects a near-expert, genuinely useful solution. This design enables a uniform evaluation framework across the diverse tasks in the suite. We use overall task success rate (i.e. the fraction of benchmark tasks successfully resolved by an

[1]Our choice of a terminal interface is motivated by evaluation rather than deployment. In practical clinical systems, agents could operate in the backend through a terminal or execution environment while presenting their outputs through clinician-facing graphical interfaces.

agent) as the primary metric for measuring overall agent progress and for comparing between tasks. Task-specific metrics, such as F1 and recall, are reported alongside task success rate to quantify partial progress and provide additional insight into agent performance.

3.4 Post-creation checks.

Before a task is added to the suite, we run a baseline sweep across multiple frontier agents and inspect the trajectories. We *manually and with LLM assistance* review the agent transcripts to catch two failure modes: (i) the agent finding an unintended shortcut (e.g. a leaked evaluation script into agent's container) which sends the task back for redesign; and (ii) genuine ambiguity in the task where multiple *reasonable* agent solutions or alternative valid gold labels are possible, which also sends the task back. A final filter is applied: we *discard tasks every agent solves trivially* and we keep only those where the success-rate gap between agents is informative.

4 Baselines and Results

4.1 Overall Performance

We evaluate 10 frontier agents from two model families (GPT-5 series [30, 29, 28] and Anthropic's Claude series [4, 3]) covering 3 harnesses (Codex [27], Claude Code [2] and Copilot-CLI [10]) with xhigh reasoning effort with disabled web browsing capabilities . Figure 4 summarizes the headline result: Across 3 attempts on 54 tasks (i.e 162 trials), the best agent (Codex GPT-5.5) achieves the task success rate of only 42% (meaning that only 42% of tasks are consistently solved), leaving ample room for improvement and suggesting that HealthAgentBench is a challenging testbed for current models. Codex GPT-5.5 is followed by Copilot CLI running Opus-4.8 and GPT-5.5 (36% and 35%), then the best native Claude Code agent, Opus-4.8 (32%). Smaller or older model families trail behind, with the weakest agent (Codex GPT-5.4-mini) achieving only 16% task success rate. Overall, the benchmark reveals a clear and steady improvement across successive model generations while remaining far from saturation.

We also show that agent harness can significantly influence performance. For example, GPT-5.5's task success rate drops from 42% under Codex to 35% under Copilot CLI, whereas Opus-4.8 improves from 32% under Claude Code to 36% under Copilot CLI. A likely reason is that Copilot CLI uses a multi-agent architecture, where the primary model delegates sub-tasks to auxiliary agents running on fixed model backends. For example, a "Copilot GPT-5.5" system is not a pure GPT-5.5 agent, but can call subagents from Opus, Sonnet and other other hardcoded models.

Beyond task success, we evaluate the cost[2] and wall-clock time of each agent. The best-performing agents are not necessarily the most expensive or the slowest. Strikingly, the costliest and slowest sweeps both belong to Claude Code agents (Opus-4.7 is the most expensive at $4.8 per task, and Sonnet-4.6 is the slowest with 24 min per task), yet all of them score well below codex GPT-5.5, which is both cheaper and faster. Plotting each agent's task success rate against the dollar cost of a full sweep over the suite (Figure 8 in Appendix C), the Pareto frontier is traced entirely by GPT-5 agents: GPT-5.4-mini, GPT-5.3, GPT-5.4, Copilot GPT-5.5, and Codex GPT-5.5 form the cost-optimal curve, while every Claude Code agent is matched or beaten on success rate by a cheaper GPT-5 run. Qualitatively, two behaviors drive up the cost of the Claude Code models: they tend to spawn many sub-processes within a run, and they are markedly more "chatty" and elaborate in their responses. For example, while Claude Code Sonnet-4.6 have significantly cheaper

[2]costs are extracted either from the providers' own billed costs or calculated using public API pricing by Harbor Framework.

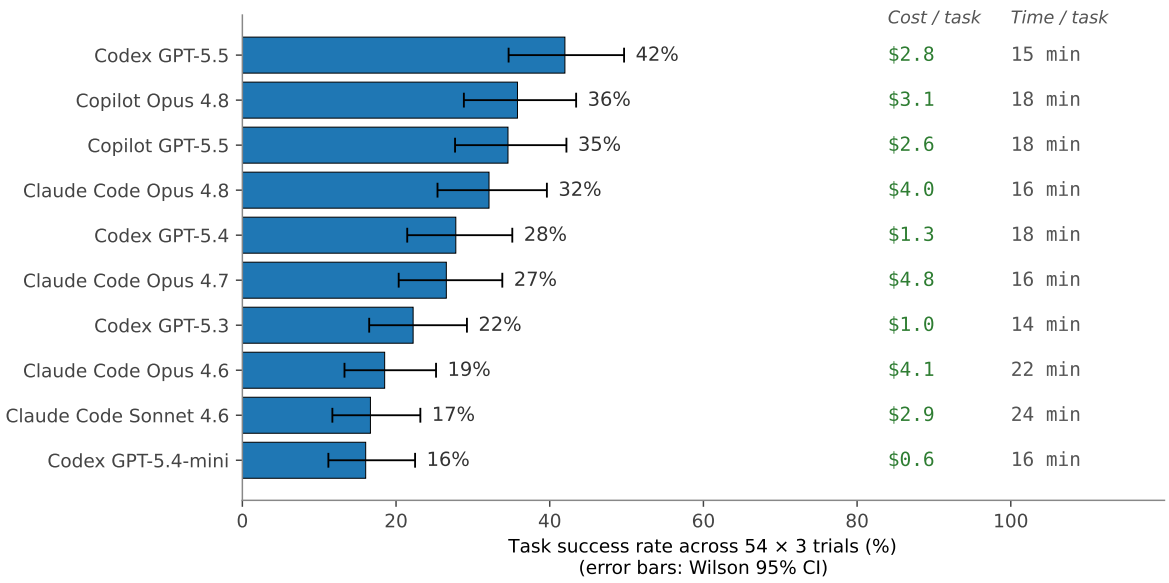


Figure 4: **Pooled task success rate of ten frontier agents on HealthAgentBench.** Each bar is the fraction of tasks the agent succeeds on, pooled across 3×54 agent trials, and error bars are Wilson 95% confidence intervals. The two right-hand columns report each agent's mean cost (USD) and mean wall-clock time (minutes) per task attempt, pooled over all trials (trial-weighted).

per-token cost, they produce around 2× as many tokens per task as the best codex GPT-5.5 run (44k vs. 20k output tokens per trial), therefore consuming more compute and costing more overall.

4.2 Breaking Down Performance by Task Categories

Figure 5 decomposes performance cell-by-cell for each task category, encoding success rate, wall clock time (a proxy for effort), and costs. While Codex GPT-5.5 achieves the best overall performance, the task breakdown shows a more nuanced picture. There is no agent that is universally the best choice across all tasks, and the difficulty of the task categories varies. Beyond the success rate as the primary metrics, we also report task-specific raw scores in the appendix (Figure 9) to track partial progress on each task which shows a ranking of models that is broadly consistent with the overall task success rate.

The Win: Machine Learning AutoResearch for EHR modelling Frontier agents are markedly capable on EHR Event Modelling and EHR Format Conversion, which replicate the research workflow of building machine learning pipelines over tabular clinical data. On EHR Format Conversion, the ETL-construction benchmark, the agent is tasked with building a data pipeline that extracts and transforms medication records from raw EHR data. On this benchmark, nine of the 10 agents reach a perfect 100% task success rate at relatively low cost and with short execution times. Only Opus-4.6 fails the task, as it consistently drops a demographic field from the output transformed data. EHR Event Modelling assesses the full pipeline of building a machine learning pipeline from EHR data for event prediction, and is thus a more difficult task. The success criterion is to reach human-engineered feature engineering baseline performance within one hour time budget. Claude Code Opus-4.7 leads at 78% with codex GPT-5.5 close behind at 72%, while most of the agents can achieve over 50% success rate, indicating that the end-to-end pipeline-building loop

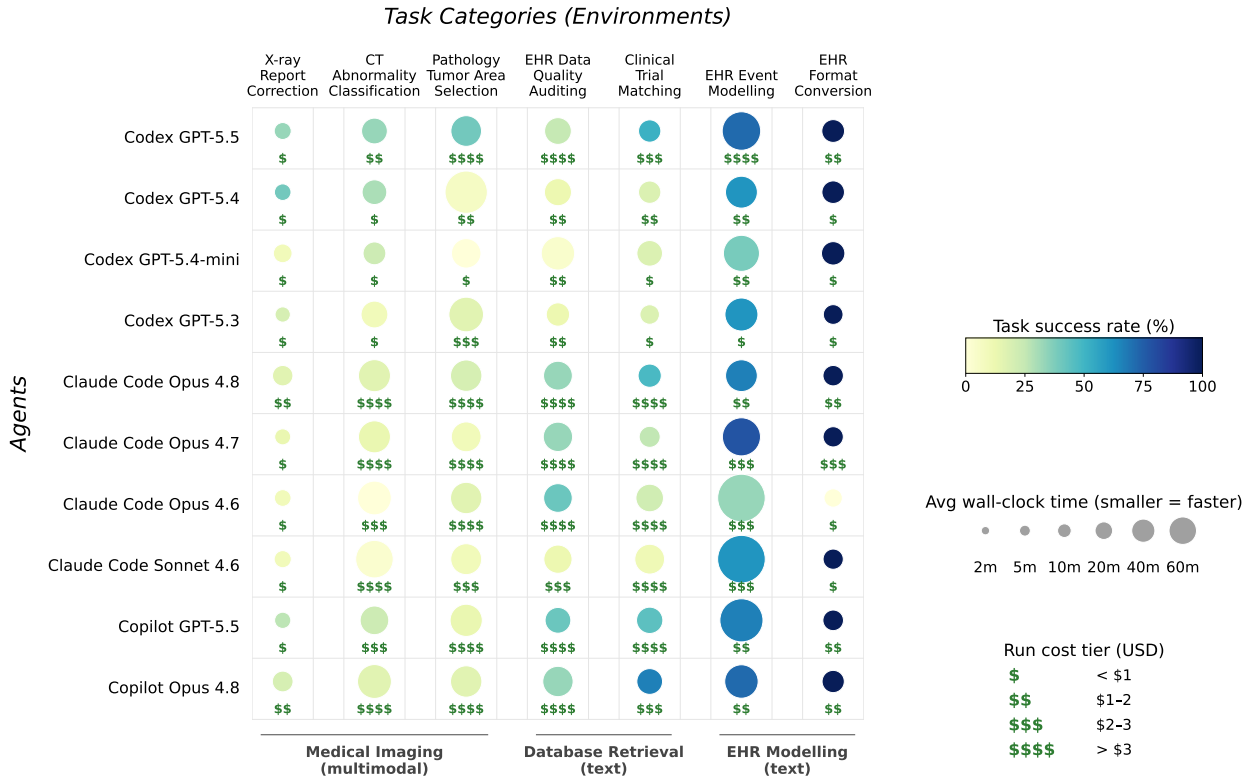


Figure 5: **Performance and efficiency of 10 frontier agents across 7 task categories.** Each task category provides a unique task environment to challenge the agents. Color denotes the per-task-category success rate, circle size shows the average wall-clock time for finishing a trial (a proxy for effort), and the $ markers denote the cost tier ($ to $$$$) of running a trial. The exact per-(agent, category) success rate, time, and cost are tabulated in Appendix D.1 (Tables 4–6). Task-specific raw scores before converting to binary success rate are also reported in Figure 9.

is broadly within reach of the current models, and the frontier agents can already competitively match human performance in performing auto research for building a ML pipeline. Notice that the time to finish a trial in this task category is the longest which is because the agent is typically iterating through different model configurations and feature sets to find the optimal solution. It is also striking that the agents reach this level with only a lightweight machine learning model with feature engineering from the raw EHR data, which is a deliberate choice given it is instructed it only has 1 hour time budget, yet their per-task AUROC matches or exceeds EHRSHOT's published CLMBR neural-network-based foundation-model baseline on most of the six prediction targets (Figure 12 in Appendix D).

The Challenge: Needle in a Haystack - Full Retrieval from Complex Search Space Many healthcare applications require browsing through large volumes of data to find relevant information. The Clinical Trial Matching and EHR Data Quality Auditing task categories are both designed to test this capability in different contexts. EHR Data Quality Auditing requires the agent to identify and correct errors in an error-injected EHR dataset, which involves forming hypotheses about what might be wrong and systematically checking for those errors. The challenge of this task is the complex search space which involves 8 tables consisting of more than 800k rows. To correctly

identify all the errors is non-trivial, and the agents struggle to do so. None of the agents achieve more than 50% task success rate (the best, Claude Code Opus-4.6, reaches 42%), and the codex gpt models other than gpt-5.5 score at or below 12.5%. This task category comprises eight tasks that are designed to disentangle two key challenges: search space and compositional complexity. To study search space, we compare four tasks in which agents receive no hints about where errors are located with four otherwise identical tasks that provide clues about the affected table and a more detailed description of the injected error. Agents perform substantially better when such hints are available, indicating that navigating large search spaces is a major bottleneck for this task category (Figure 10 in Appendix D.3). To study compositional complexity, two tasks require agents to identify all injected errors in a single run, whereas the remaining six each target a single error type. Performance drops markedly when agents must detect all errors simultaneously, compared with aggregating the results of separate runs targeting individual error types. This finding suggests that multi-step reasoning and increased task compositionality constitute a second major bottleneck (Figure 11 in Appendix D.3).

Clinical Trial Matching presents a different retrieval challenge. Rather than searching a large relational database as in EHR Data Quality Auditing, the agent must identify eligible clinical trials for a patient from a corpus of roughly 400 free-text trial protocols. Although the search space is smaller, it is more difficult to navigate because both the patient profile and eligibility criteria are expressed in unstructured text. The key challenge is therefore to efficiently triage the corpus before performing fine-grained eligibility reasoning. A clear performance gap emerges between the leading frontier agents and the rest of the evaluated systems, although even the best agents remain far from saturating the benchmark. Copilot CLI with Opus-4.8 achieves the highest task success rate (67%), followed by Codex GPT-5.5 (52%), Claude Code Opus-4.8 (48%), and Copilot CLI with GPT-5.5 (44%), whereas all remaining agents achieve only 11%–26%. Inspection of the best agent's success trajectory (Copilot CLI Opus-4.8; Appendix F) reveals an effective strategy: it first triages the corpus with a lightweight retrieval script, then fans the eligibility decision out to parallel subagents, each adjudicating a disjoint batch of candidate trials against a structured patient profile before centrally re-verifying the borderline trials' criteria rather than relying on surface keyword matching. Looking directly at the recall performance further highlights the difficulty of the benchmark (Figure 9). The strongest agents retrieve most eligible trials (recall $\approx$ 0.8–0.9), yet achieving a perfect recall of 1.0 as required to pass the task remains challenging, as it demands a more thorough search and an exact understanding of the eligibility criteria and patient profile.

The Challenge: Medical Imaging and the emerging capabilities from Codex GPT-5.5
Medical imaging tasks (CT Abnormality Classification, X-ray Report Correction and Pathology Tumor Area Selection) pose significant challenges for every agent. Averaged across all codex and claude code agents, the mean success rate is only 17% on the imaging tasks versus 49% on the text tasks. The strongest agent on imaging (Codex GPT-5.5) averages only $\sim$35% across the three imaging tasks and no agent exceeds 40% on any single imaging task category. This is expected as the agents are required to process specialized medical modalities and need to deal with excessive input size (for example a ct volume typically consists of hundreds of slices, and a pathology slide can be gigapixels in size). The agent needs to find a clever way to decompose the tasks into manageable chunks and design a good strategy to divide and conquer. While most of the agents struggle on these tasks, the codex GPT family shows an overall better performance than the Claude Code agents, and GPT-5.5 in particular shows a marked improvement. For example, on the Pathology Tumor Area Selection category GPT-5.5 succeeds on 40% of the tasks, well clear of every other agent (the best Claude Code agents, Opus-4.8 and Opus-4.6, reach only 20% and 17%). Inspecting

its trajectory reveals a human-like, pathologist-style workflow for reviewing the whole slide: because it cannot view the gigapixel image directly, it reads the slide through the OpenSlide image pyramid and renders downsampled overviews and contact sheets aligned to the scoring grid, distinguishes carcinoma from benign lymphoid tissue *morphologically* rather than by a simple stain threshold, and then refines the tumor boundary tile-by-tile at full resolution, running an explicit "outside pass" over unclaimed high-tissue tiles to recover any missed regions before submitting (Appendix F). Figure 13 shows one such successful slide, where GPT-5.5 recovers the entire tumor region with only two spurious tiles.

The divide between the Codex GPT family and the Claude Code family on imaging is also apparent (Figure 6): the Codex agents reach a 22% mean success rate on imaging against only 12% for the Claude Code agents, whereas on the four text tasks the two families are comparable (50% vs. 48%). The two frontier Codex agents (GPT-5.5 and GPT-5.4) lead every Claude Code agent on imaging. This edge holds even though the Claude Code agents often take more turns and spend more per run, reinforcing that on these perception-heavy tasks the Codex models extract more signal per unit of effort. This likely drives the overall performance gain of Codex GPT-5.5 over Claude Code Opus-4.8 across the whole task suite. Overall, while there are emerging promises, the challenge from the imaging tasks is still significant, and the best agents are still far from clinically useful performance. Closing the gap to clinically useful accuracy will likely require tool augmentation or specialized vision backends rather than larger general-purpose agents alone.

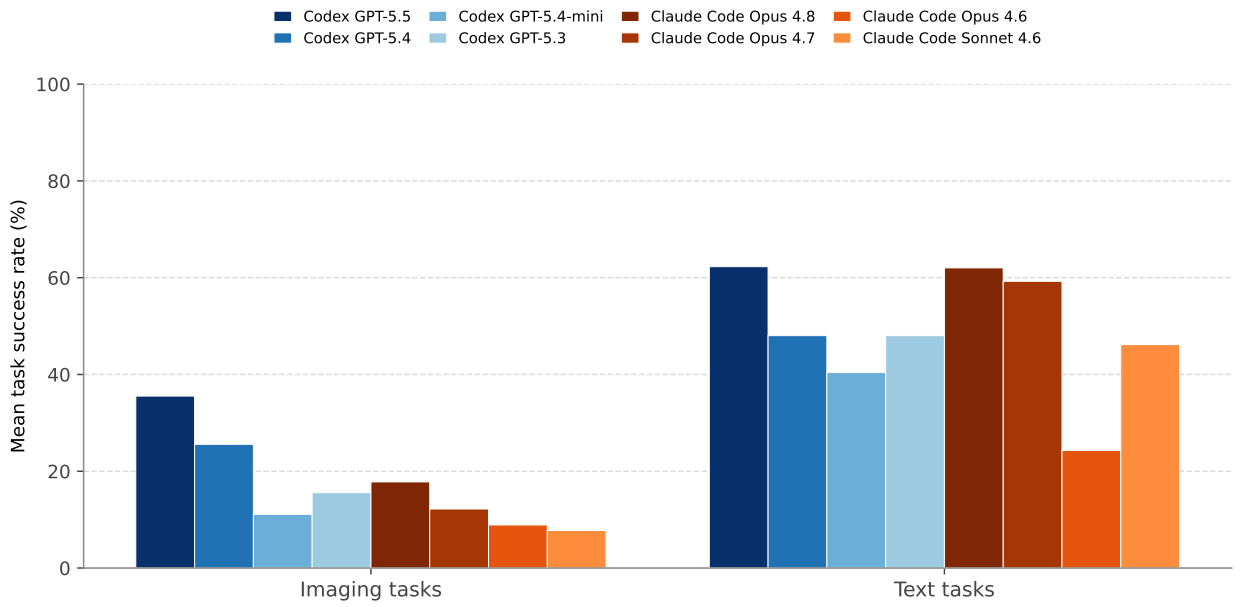


Figure 6: **Imaging vs. text task performance for Codex and Claude Code agents.** Each bar is an agent's mean task success rate aggregated over the tasks in a modality group: *imaging* tasks pool X-ray Report Correction, CT Abnormality Classification, and Pathology Tumor Area Selection, while *text* tasks pool EHR Format Conversion, EHR Data Quality Auditing, EHR Event Modelling, and Clinical Trial Matching. Imaging tasks are overall significantly harder than text tasks and the two frontier Codex agents (blues), GPT-5.5 and GPT-5.4, lead every Claude Code agent (oranges). On text tasks the ordering is mixed and the best agents from both Codex and Claude Code families are comparable.

5 Conclusion

We introduced HealthAgentBench, a unified, agent-native benchmark suite for evaluating healthcare AI agents across diverse data modalities and clinical workflows spanning the patient journey. The benchmark remains highly challenging for current frontier agents: even the strongest agent, Codex GPT-5.5, achieves a mean success rate of only 42% across the suite, leaving substantial room for future progress. Our evaluation of 10 frontier agents also reveals important strengths and limitations of current systems, highlighting persistent challenges in medical imaging, large search spaces, and complex compositional reasoning. While HealthAgentBench offers a practical interface for measuring and tracking progress in agentic AI for healthcare, we acknowledge that the benchmark is not exhaustive for all aspects of healthcare agents, and we encourage future work to add additional tasks and modalities to better capture the full spectrum of agentic AI capabilities in healthcare, and we hope it becomes a useful community resource for evaluating future healthcare agents.

References

[1] Mohammad Almansoori, Komal Kumar, and Hisham Cholakkal. Self-evolving multi-agent simulations for realistic clinical interactions, 2025. URL https://arxiv.org/abs/2503.22678.

[2] Anthropic. Claude code documentation, 2025. URL https://code.claude.com/docs.

[3] Anthropic. Claude opus system card, 2026. URL https://www.anthropic.com/claude/opus.

[4] Anthropic. Claude sonnet system card, 2026. URL https://www.anthropic.com/claude/sonnet.

[5] Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025. URL https://doi.org/10.48550/arXiv.2505.08775.

[6] Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. HealthBench: Evaluating Large Language Models Towards Improved Human Health, 2025. URL https://doi.org/10.48550/arXiv.2505.08775.

[7] Suhana Bedi, Ryan Welch, Ethan Steinberg, Michael Wornow, Taeil Matthew Kim, Haroun Ahmed, Peter Sterling, Bravim Purohit, Qurat Akram, Angelic Acosta, et al. Healthadmin-bench: Evaluating computer-use agents on healthcare administration tasks. *arXiv preprint arXiv:2604.09937*, 2026. URL https://doi.org/10.48550/arXiv.2604.09937.

[8] Zhihao Fan, Jialong Tang, Wei Chen, Siyuan Wang, Zhongyu Wei, Jun Xi, Fei Huang, and Jingren Zhou. Ai hospital: Benchmarking large language models in a multi-agent medical interaction simulator, 2024. URL https://arxiv.org/abs/2402.09742.

[9] Yifan Gao, Haoyue Li, Feng Yuan, Xin Gao, Weiran Huang, and Xiaosong Wang. Camyla: Scaling autonomous research in medical image segmentation, 2026. URL https://arxiv.org/abs/2604.10696.

[10] GitHub. Github copilot cli, 2025. URL https://github.com/features/copilot/cli.

[11] Ibrahim Ethem Hamamci, Sezgin Er, Chenyu Wang, Furkan Almas, Ayse Gulnihan Simsek, Sevval Nil Esirgun, Irem Dogan, Omer Faruk Durugol, Benjamin Hou, Suprosanna Shit, Weicheng Dai, Murong Xu, Hadrien Reynaud, Muhammed Furkan Dasdelen, Bastian Wittmann, Tamaz Amiranashvili, Enis Simsar, Mehmet Simsar, Emine Bensu Erdemir, Abdullah Alanbay, Anjany Sekuboyina, Berkan Lafci, Ahmet Kaplan, Zhiyong Lu, Malgorzata Polacin, Bernhard Kainz, Christian Bluethgen, Kayhan Batmanghelich, Mehmet Kemal Ozdemir, and Bjoern Menze. Generalist Foundation Models from a Multimodal Dataset for 3D Computed Tomography. *Nature Biomedical Engineering*, 2026. URL https://doi.org/10.1038/s41551-025-01599-y.

[12] Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, January 2026. URL https://github.com/laude-institute/harbor.

[13] Rebecca Soskin Hicks, Mikhail Trofimov, Dominick Lim, Rahul K. Arora, Foivos Tsimpourlas, Preston Bowman, Michael Sharman, Chi Tong, Kavin Karthik, Arnav Dugar, Akshay Jagadeesh, Khaled Saab, Johannes Heidecke, Ashley Alexander, Nate Gross, and Karan Singhal. Healthbench professional: Evaluating large language models on real clinician chats, 2026. URL https://arxiv.org/abs/2604.27470.

[14] Yixing Jiang, Kameron C. Black, Gloria Geng, Danny Park, James Zou, Andrew Y. Ng, and Jonathan H. Chen. MedAgentBench: A Virtual EHR Environment to Benchmark Medical LLM Agents. *NEJM AI*, 2, 2025. URL https://doi.org/10.1056/AIdbp2500144.

[15] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021. URL https://doi.org/10.3390/app11146421.

[16] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV Clinical Database Demo. *PhysioNet*, January 2023. doi: 10.13026/dp1f-ex47. URL https://doi.org/10.13026/dp1f-ex47. Version 2.2.

[17] Alistair Johnson, Matthew Lungren, Yifan Peng, Zhiyong Lu, Roger Mark, Seth Berkowitz, and Steven Horng. MIMIC-CXR-JPG - chest radiographs with structured labels. *PhysioNet*, March 2024. doi: 10.13026/jsn5-t979. URL https://doi.org/10.13026/jsn5-t979. Version 2.1.0.

[18] Alistair Johnson, Tom Pollard, Roger Mark, Seth Berkowitz, and Steven Horng. MIMIC-CXR Database. *PhysioNet*, July 2024. doi: 10.13026/4jqj-jw95. URL https://doi.org/10.13026/4jqj-jw95. Version 2.1.0.

[19] Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1, 2023. doi: 10.1038/s41597-022-01899-x.

[20] Yusheng Liao, Shuyang Jiang, Yanfeng Wang, and Yu Wang. ReflecTool: Towards Reflection-Aware Tool-Augmented Clinical Agents. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL https://doi.org/10.18653/v1/2025.acl-long.663.

[21] Junqi Liu, Selena Song, Yuhan Wang, Jiawei Mao, Hardy Chen, Xiaoke Huang, Tianhao Qi, Pengfei Guo, Yucheng Tang, Yufan He, Can Zhao, Andriy Myronenko, Dong Yang, Daguang Xu, and Yuyin Zhou. Automedbench: Towards medical autoresearch with agentic ai models, 2026.

[22] Ruoqi Liu, Imran Q. Mohiuddin, Austin J. Schoeffler, Kavita Renduchintala, Ashwin Nayak, Prasantha L. Vemu, Shivam C. Vedak, Kameron C. Black, John L. Havlik, Isaac Ogunmola, Stephen P. Ma, Roopa Dhatt, and Jonathan H. Chen. Physicianbench: Evaluating llm agents in real-world ehr environments, 2026. URL https://arxiv.org/abs/2605.02240.

[23] Matthew McDermott and Justin Xu. MIMIC-IV MEDS: An ETL pipeline to extract MIMIC-IV data into the MEDS format. https://github.com/Medical-Event-Data-Standard/MIMIC_IV_MEDS, May 2025. Software, MIT license.

[24] Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

[25] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for General AI Assistants, 2023. URL https://doi.org/10.48550/arXiv.2311.12983.

[26] Harsha Nori, Mayank Daswani, Christopher Kelly, Scott Lundberg, Marco Tulio Ribeiro, Marc Wilson, Xiaoxuan Liu, Viknesh Sounderajah, Jonathan Carlson, Matthew P Lungren, Bay Gross, Peter Hames, Mustafa Suleyman, Dominic King, and Eric Horvitz. Sequential diagnosis with language models, 2025. URL https://arxiv.org/abs/2506.22405.

[27] OpenAI. Openai codex cli, 2025. URL https://github.com/openai/codex.

[28] OpenAI. Gpt-5.3 codex, 2026. URL https://openai.com/index/introducing-gpt-5-3-codex/.

[29] OpenAI. Gpt-5.4, 2026. URL https://openai.com/index/gpt-5-4.

[30] OpenAI. Introducing gpt-5.5, 2026. URL https://openai.com/index/introducing-gpt-5-5.

[31] Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor. Agentclinic: a multimodal agent benchmark to evaluate ai in simulated clinical environments, 2025. URL https://arxiv.org/abs/2405.07960.

[32] Ian Soboroff. Overview of trec 2021. In *TREC*, 2021.

[33] Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason A. Fries, and Nigam H. Shah. EHRSHOT: An EHR Benchmark for Few-Shot Evaluation of Foundation Models, 2023. URL https://doi.org/10.48550/arXiv.2307.02028.

[34] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments, 2024. URL https://doi.org/10.48550/arXiv.2404.07972.

[35] Ran Xu, Yuchen Zhuang, Yishan Zhong, Yue Yu, Zifeng Wang, Xiangru Tang, Hang Wu, May Dongmei Wang, Peifeng Ruan, Donghan Yang, Tao Wang, Guanghua Xiao, Xin Liu, Carl Yang, Yang Xie, and Wenqi Shi. Medagentgym: A scalable agentic training environment for code-centric reasoning in biomedical data science. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jHDZEUgS4r.

[36] Juan Manuel Zambrano Chaves, Shih-Cheng Huang, Yanbo Xu, Hanwen Xu, Naoto Usuyama, Sheng Zhang, Fei Wang, Yujia Xie, Mahmoud Khademi, Ziyi Yang, et al. A clinically accessible small multimodal radiology model and evaluation metric for chest x-ray findings. *Nature Communications*, 16(1):3108, 2025.

[37] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents, 2024. URL https://doi.org/10.48550/arXiv.2307.13854.

[38] Yakun Zhu, Zhongzhen Huang, Qianhan Feng, Linjie Mu, Yannian Gu, Shaoting Zhang, Qi Dou, and Xiaofan Zhang. CP-Env: Evaluating Large Language Models on Clinical Pathways in a Controllable Hospital Environment, 2025. URL https://doi.org/10.48550/arXiv.2512.10206.

[39] Yinghao Zhu, Ziyi He, Haoran Hu, Xiaochen Zheng, Xichen Zhang, Zixiang Wang, Junyi Gao, Liantao Ma, and Lequan Yu. MedAgentBoard: Benchmarking Multi-Agent Collaboration with Conventional Methods for Diverse Medical Tasks, 2025. URL https://doi.org/10.48550/arXiv.2505.12371.

A Benchmark Comparison

Table 1 gives the full per-benchmark comparison from which the positioning of Figure 2 is derived.

Benchmark	Modality	Interaction	Workflow stage
Question answering			
MedAgentBoard [39]	text, EHR, 2D	static QA	diagnosis, comms, research, data
ClinicalAgent Bench [20]	text, EHR, 2D	static QA	diagnosis, data
Conversational / simulated clinic			
HealthBench [6]	text	dialogue / simulation	comms, diagnosis
HealthBench Professional [13]	text	dialogue / simulation	comms, diagnosis, treatment, research
MAI-DxO [26]	text	dialogue / simulation	diagnosis
AgentClinic [31]	text, EHR, 2D	dialogue / simulation	diagnosis, comms
AI Hospital [8]	text	dialogue / simulation	diagnosis, comms
MedAgentSim [1]	text, 2D	dialogue / simulation	diagnosis, comms
CP-Env [38]	text	dialogue / simulation	diagnosis, treatment, comms
Executable environment			
MedAgentBench [14]	EHR	executable	treatment, data
PhysicianBench [22]	text, EHR	executable	diagnosis, treatment, data
MedAgentGym [35]	text, EHR	executable	data, research
AutoMedBench [21]	2D, 3D, WSI	executable	research
CamylaBench [9]	2D, 3D	executable	research
HealthAdminBench [7]	GUI, EHR	executable	admin
HealthAgentBench	**text, EHR, 2D, 3D, WSI**	**executable**	**diagnosis, treatment, data, research**

Table 1: **Where HealthAgentBench sits among healthcare agent benchmarks.** HealthAgentBench row spans five modalities and four workflows. The positioning of Figure 2 is read directly off this table. *Modality* counts data the agent must *process as raw input*: **text**; **EHR** (structured/tabular records via FHIR/SQL); **2D** (2-D image—radiograph, dermatology/photo, ultrasound, fundus, blood smear); **3D** (CT/MRI volume); **WSI** (whole-slide pathology); **GUI** (computer-use web interface). *Interaction* is the interaction-realism / agent-autonomy category, the x-axis of Figure 2: **static QA**, **dialogue / simulation**, or **executable** environment. *Workflow stage* uses six buckets: **diagnosis** (clinical assessment, consultation, and reaching a diagnosis); **treatment** (treatment/management planning, prescribing, ordering care); **comms** (patient/clinician communication—dialogue, advice, and documentation); **data** (operating on health data—EHR query/order/write, ETL, data-quality auditing, record retrieval); **research** (ML/analytic pipeline building and evidence synthesis); **admin** (hospital administration and operations).

B The HealthAgentBench Benchmark Suite

HealthAgentBench is built around the Harbor execution substrate [12]: each task is packaged as a Harbor task with a standardized container environment with internet (for the agent to install dependencies etc.), agent-visible instructions, hidden test labels, and a verifier that emits both a binary success/failure reward and richer diagnostic metrics. The first release of HealthAgentBench integrates seven benchmark categories spanning EHR Format Conversion, EHR Event Modelling, EHR Data Quality Auditing, Clinical Trial Matching, X-ray Report Correction, CT Abnormality Classification, and Pathology Tumor Area Selection. Table 2 summarizes the seven task categories in the suite. The rest of this section describes each task category.

Task Category	Modality	Task type	# tasks
EHR Format Conversion	EHR (MEDS)	Pipeline customization	1
X-ray Report Correction	Imaging + text	Report correction	10
Clinical Trial Matching	Text	Eligibility matching	9
CT Abnormality Classification	3D imaging	Abnormality detection	10
Pathology Tumor Area Selection	Pathology WSI	Tumor area selection	10
EHR Event Modelling	Longitudinal EHR	Clinical event prediction	6
EHR Data Quality Auditing	Tabular EHR	Data-quality auditing	8

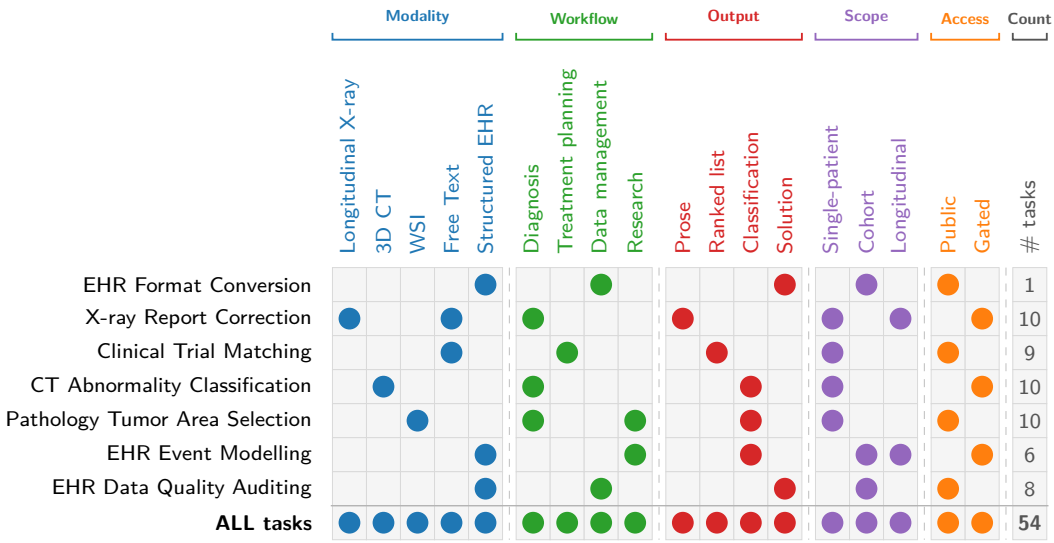
Table 2: Benchmarks in HealthAgentBench. "# tasks" is the number of tasks belong to each task category

Success criteria. Every task in HealthAgentBench reduces to a single binary success/failure outcome: a trial earns reward 1 only when it meets the task's success criterion, and 0 otherwise. The criteria are defined per task category based on different underlying metrics (verifier checks, error counts, accuracy, F1, recall, and AUROC). The task success rates reported in Section 4 aggregate these binary outcomes across tasks. Table 3 summarizes the success criteria for every task category; the per-task-category *Scoring* paragraphs later in this appendix give the full details, and the per-task fine-grained metrics are reported in Appendix D.

Task Category	Success metric	Success criteria (reward = 1)
EHR Format Conversion	Verifier checks	All config and output checks pass
X-ray Report Correction	Significant errors	$\geq 3/5$ judge votes report zero clinically-significant errors
Clinical Trial Matching	Recall@top-50	All eligible trials ranked within the 50 most-confident picks (recall = 1.0)
CT Abnormality Classification	Per-label accuracy	Every abnormality label correct (accuracy = 1.0)
Pathology Tumor Area Selection	Tile-level F1	Tile-F1 against the gold tumor mask $\geq$ 0.90
EHR Event Modelling	Test AUROC	AUROC $\geq$ the count+GBM baseline
EHR Data Quality Auditing	Cluster recall	Every injected error cluster flagged (recall = 1.0, precision $\geq$ 0.01)

Table 3: Success criteria for each task category. A trial earns reward 1 iff its outcome meets the criteria in the last column. The "Success metric" is the underlying continuous metric the criteria thresholds; these per-task metrics are charted in Figure 9.

Coverage across evaluation axes. Beyond the per-task-category summary of Table 2, Figure 7 maps the seven task categories onto six orthogonal evaluation axes: input *modality* (2-D radiograph, 3-D CT, whole-slide pathology, free text, structured EHR), *clinical workflow stage* (diagnosis, treatment planning, data management, research), agent *output shape* (prose report, ranked list, classification, engineered solution), *patient and temporal scope* (single record vs. cohort, cross-sectional vs. longitudinal), *data-access regime* (local vs. credentialed on-the-fly download). A filled cell indicates the task category exercises that dimension along the axes. Most task categories occupy several dimensions within an axis at once—X-ray Report Correction, for instance, is both imaging and text and both single-patient and longitudinal. The "All tasks" row at the bottom of the figure shows that HealthAgentBench as a whole spans all dimensions.



Filled dot = the task categories covers that dimension. Color encodes axis. Last column counts the number of tasks in each task category; the ALL-tasks row sums across the suite.

Figure 7: **Coverage of HealthAgentBench tasks across five evaluation axes** Rows are seven task categories; columns are dimension values within five axes (modality, clinical workflow stage, output shape, patient/temporal scope, and data access regime); WSI denotes whole-slide (pathology) imaging. A filled dot indicates the task category covers that dimension of the axis. The rightmost column reports the number of tasks each task category contains. The bottom "ALL tasks" row shows that every dimension value across every axis is covered by at least one task category in the suite, and reports the total task count (54). Most task categories span multiple axes simultaneously (e.g., X-ray Report Correction is multimodal with both X-ray and text inputs, and is both single-patient and longitudinal), giving a single HealthAgentBench sweep broad evaluative reach.

C Cost vs. performance trade-off

Section 4 notes that the best-performing agents are not the most expensive. Figure 8 makes this explicit by plotting each agent's task success rate against the cost of running it over the full HealthAgentBench suite. The x-axis is the total USD cost of one full sweep over the suite on a log scale, and the y-axis is the pooled, trial-weighted task success rate (identical to Figure 4). The Pareto frontier is traced entirely by GPT-5 agents: GPT-5.4-mini, GPT-5.3, GPT-5.4, Copilot GPT-5.5, and Codex GPT-5.5 form the cost-optimal curve, while every Claude Code agent lies below-and-right of it (matched or beaten on success rate by a cheaper GPT-5 run).

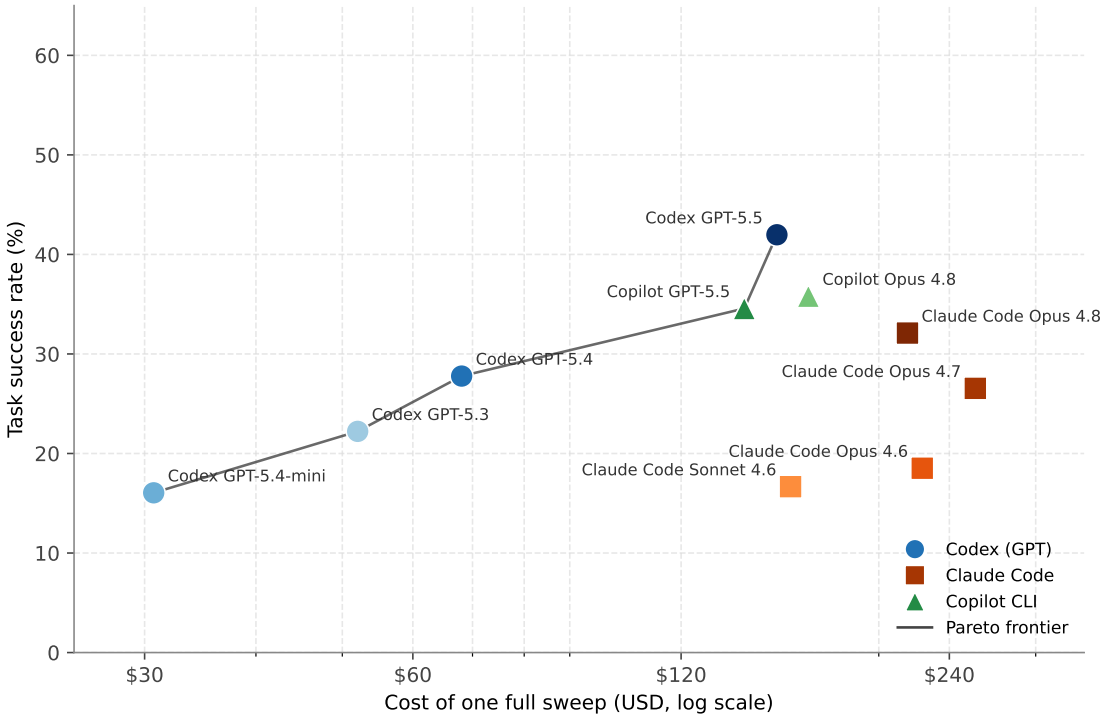


Figure 8: **Cost vs. success rate on HealthAgentBench.** Each point is an agent; the x-axis is the total USD cost of one full sweep over the suite of 54 tasks and the y-axis is the pooled task success rate (trial-weighted, as in Figure 4). The Pareto frontier (line) is traced entirely by GPT-5 agents (Codex and Copilot CLI); every Claude Code agent lies below-and-right of it (more expensive for equal-or-lower success rate).

D Per-Category Performance Breakdowns

This appendix collects finer-grained, per-category performance breakdowns that complement the aggregate results in Section 4.

D.1 Per-category results: success rate, time, and cost

Figure 5 summarizes, for every (agent, task category) cell, three quantities at once—via colour, circle size, and the $ tier. Tables 4–6 report the exact values behind that figure so each cell can be read precisely. Table 4 gives the binary *task success rate* (%); Table 5 the mean *wall-clock time* per

trial (minutes), a proxy for effort; and Table 6 the mean *cost* per trial (USD). Every cell pools the three attempts on each instance in the category at xhigh reasoning effort, and the ten agents span the three harnesses (Codex, Claude Code, and Copilot CLI).

Agent	X-ray	CT	Tumor	Trial	DQ	EHR	MEDS
Codex GPT-5.5	33	33	40	52	25	72	100
Codex GPT-5.4	40	30	7	18	12	61	100
Codex GPT-5.4-mini	10	23	0	18	4	39	100
Codex GPT-5.3	20	10	17	18	12	61	100
Claude Code Opus 4.8	17	17	20	48	33	67	100
Claude Code Opus 4.7	13	13	10	26	33	78	100
Claude Code Opus 4.6	10	0	17	22	42	33	0
Claude Code Sonnet 4.6	10	3	10	11	12	61	100
Copilot GPT-5.5	27	23	13	44	42	67	100
Copilot Opus 4.8	20	17	17	67	33	72	100

Table 4: **Task success rate (%)** per agent and task category. Columns: X-ray Report Correction, CT Abnormality Classification, Pathology Tumor Area Selection, Clinical Trial Matching, EHR Data Quality Auditing, EHR Event Modelling, EHR Format Conversion.

Agent	X-ray	CT	Tumor	Trial	DQ	EHR	MEDS
Codex GPT-5.5	4	13	21	9	14	37	9
Codex GPT-5.4	3	12	46	9	15	23	9
Codex GPT-5.4-mini	5	9	19	13	26	31	10
Codex GPT-5.3	3	14	28	6	10	25	6
Claude Code Opus 4.8	7	23	22	10	18	23	7
Claude Code Opus 4.7	3	23	19	7	18	36	6
Claude Code Opus 4.6	3	26	22	16	17	61	5
Claude Code Sonnet 4.6	4	34	21	19	17	61	6
Copilot GPT-5.5	3	17	23	14	13	48	7
Copilot Opus 4.8	7	27	22	13	20	26	9

Table 5: **Mean wall-clock time per trial (minutes)** per agent and task category.

Agent	X-ray	CT	Tumor	Trial	DQ	EHR	MEDS
Codex GPT-5.5	0.9	2.0	5.2	2.4	3.8	3.3	1.1
Codex GPT-5.4	0.4	0.8	2.0	1.0	1.9	1.7	0.8
Codex GPT-5.4-mini	0.2	0.2	0.6	0.5	1.0	1.1	0.3
Codex GPT-5.3	0.2	0.8	2.0	0.7	1.3	0.8	0.4
Claude Code Opus 4.8	1.1	4.7	4.6	7.7	3.8	1.6	1.9
Claude Code Opus 4.7	0.5	6.9	6.1	5.5	6.6	2.9	2.3
Claude Code Opus 4.6	0.3	2.8	6.0	10.1	3.5	2.0	0.9
Claude Code Sonnet 4.6	0.2	3.1	2.5	7.6	2.2	2.4	0.7
Copilot GPT-5.5	0.6	2.2	4.3	3.1	3.8	1.8	1.1
Copilot Opus 4.8	1.4	4.2	4.0	2.8	4.0	1.9	1.9

Table 6: **Mean cost per trial (USD)** per agent and task category.

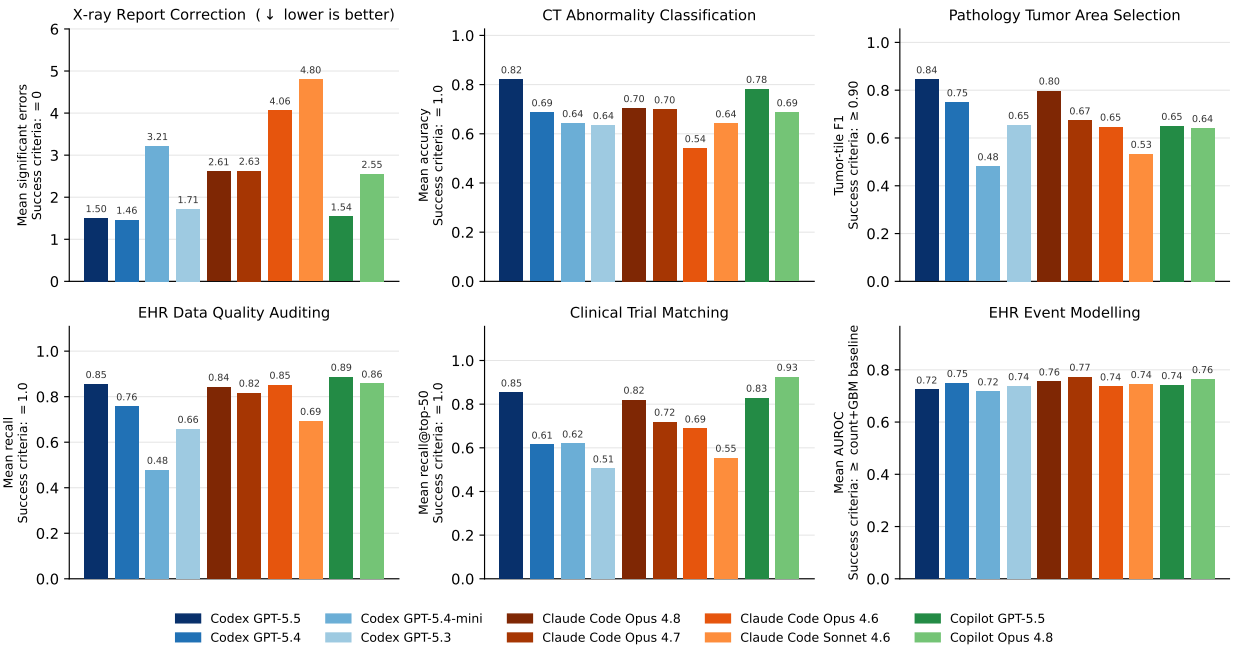


Figure 9: **Category-specific scores per agent.** One panel per benchmark; each bar is an agent. Each panel uses its own y-axis because the metrics live on different scales. For X-ray Report Correction the metric is a mean error count, so *lower is better*; all other panels are higher-is-better.

D.2 Task-specific scores

The headline task success rate reduces each task to a binary success/failure. Figure 9 reports the *task-specific* metric before converting to binary score except EHR Format Conversion, whose only score is the binary reward: mean accuracy for CT Abnormality Classification, tumor-tile F1 for Pathology Tumor Area Selection, mean recall for EHR Data Quality Auditing, mean recall@top-50 for Clinical Trial Matching, mean AUROC for EHR Event Modelling, and the mean count of significant errors for X-ray Report Correction (where lower is better). Each panel's y-axis also notes the task's success criterion.

D.3 EHR Data Quality Auditing: the cost of a large search space

Two further breakdowns isolate *why* EHR Data Quality Auditing is hard, and both point to the size of the search space rather than to any single difficult error type.

There are eight tasks in EHR Data Quality Auditing covering four scenarios: impossible clinical values, demographic conflicts, conflicting/duplicate cross-table records, and a combined task mixing all three. Each of the four scenarios in EHR Data Quality Auditing ships in two instruction variants over same data, labels, and scoring: a base task that names only the high-level error families, and a *clue* variant whose instructions additionally disclose the specific injected sub-types and name the table(s) to inspect for each. Figure 10 compares the two. Telling the agent *where* to look improves recall for every agent with the strongest agents reaching near-perfect recall once the table is disclosed (Claude Code Opus-4.8 climbs to 0.97). That the same data becomes markedly more solvable when the search is narrowed confirms that locating the errors within eight tables and >800k rows is the dominant bottleneck.

A complementary view asks whether targeting one error type at a time beats hunting for all of

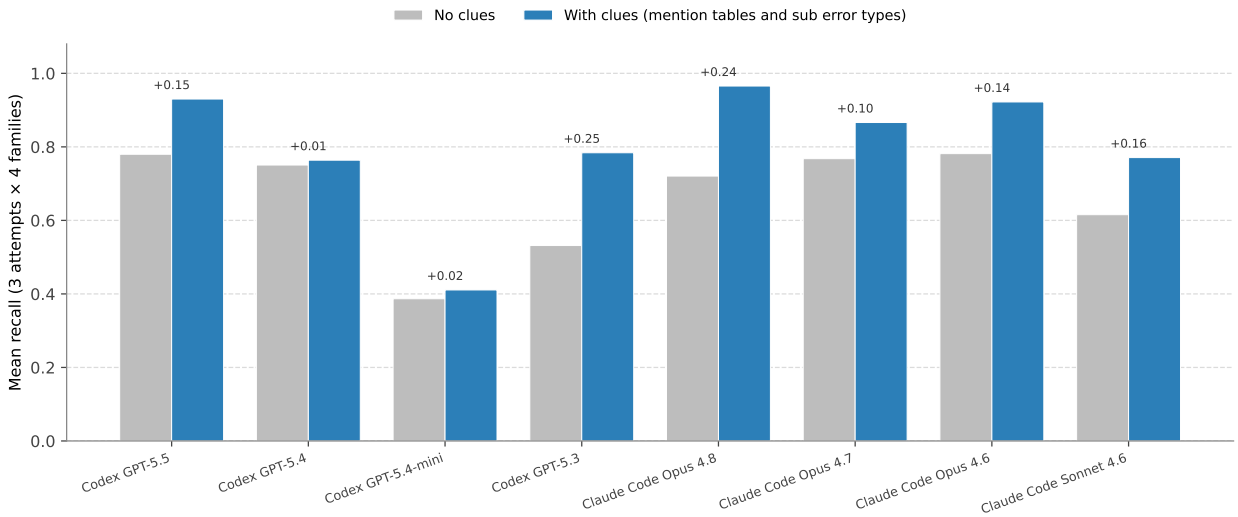


Figure 10: **Recall performance on EHR Data Quality Auditing with and without search clues.** For each agent, the left bar is mean recall on the four base tasks (only the high-level error families are named) and the right bar is mean recall on the four *clue* variants, which additionally disclose the injected sub-types and which table(s) to inspect. Each bar averages three attempts on each of the four error families (12 trials); the two variants share identical data, labels, and scoring, so the gap (annotated above each pair) isolates the value of narrowing the search. Disclosing where to look helps every agent and lifts the strongest to near-perfect recall, indicating that the size of the search space is the primary bottleneck.

them at once. The combined task injects all three error families into a single corpus whose label set is exactly the union of the three single-error tasks, so the only difference is whether the agent triages everything in one pass or in three focused passes. Figure 11 contrasts the recall an agent achieves when the three single-error tasks are run separately and their catches pooled into one aggregated recall (true positives summed over the three tasks, divided by the pooled label count, averaged across the clue and no-clue variants) against its mean recall on the combined task. Decomposition helps every agent: splitting the same errors into focused passes recovers substantially more of them than confronting them all at once. This degradation under compositional load is a key challenge in EHR Data Quality Auditing.

D.4 EHR Event Modelling: AUROC vs. human engineered baseline

The EHR Event Modelling benchmark asks the agent to predict six new-onset diagnoses from a longitudinal EHR event log. Figure 12 compares the four strongest agents against two EHRSHOT reference points, both evaluated on the same first-prediction-time test cohort as the agents: the published *CLMBR* baseline (a frozen clinical language-model representation fed to a logistic-regression head) and a *count+GBM* baseline (a gradient-boosted classifier over hand-engineered count features). Given the 1 hour time budget, we use the count+GBM baseline as the pass bar as the neural-network-based CLMBR baseline is more expensive and time-consuming to run. We observe that the best agents can match or even exceed both baselines on most targets, with the exception of pancreatic cancer where the count+GBM baseline is the strongest reference.

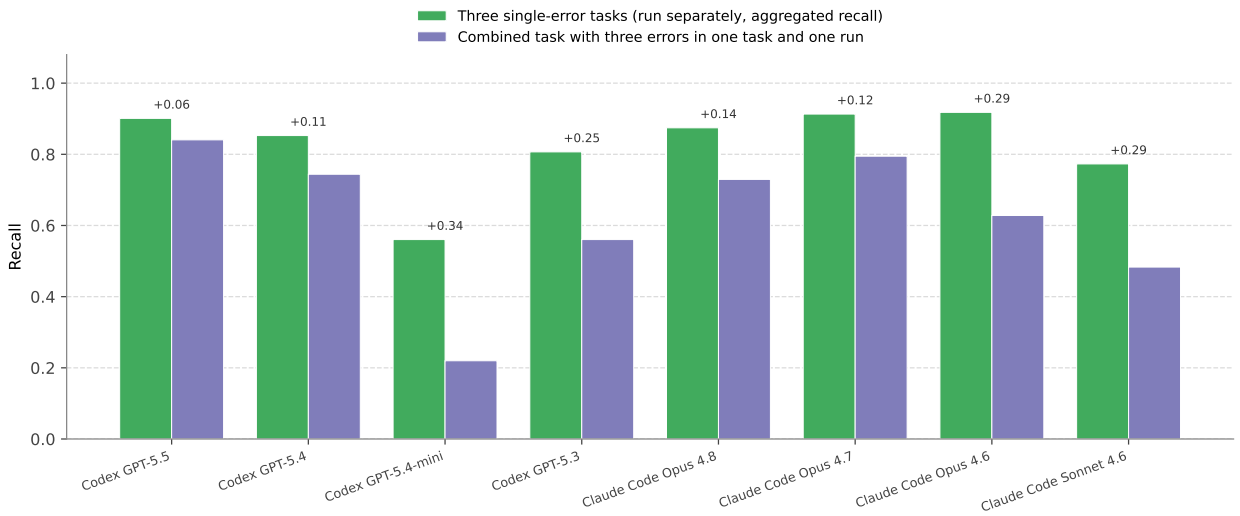


Figure 11: **Recall performance on EHR Data Quality Auditing: three single-error tasks run separately vs. the combined task.** The left bar aggregates recall across the three single-error tasks as if the agent ran them one-by-one. True positives are pooled over the three tasks: demographic conflict, impossible values, and and table inconsistencies, divided by the pooled label count, then averaged over the clue and no-clue variants. The right bar is the agent's mean recall on the combined task (clue and no-clue averaged), which injects all three families into one corpus with the same union label set. Every agent recalls more when the error families are handled separately (gap annotated above each pair), showing that compositional load makes the task more challenging in EHR Data Quality Auditing.

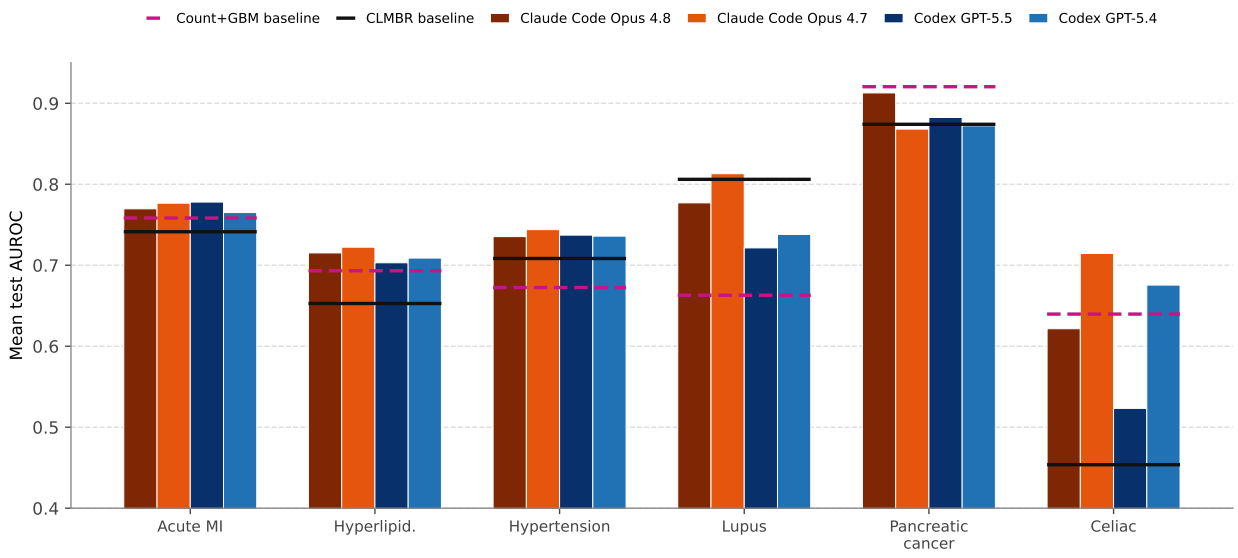


Figure 12: **Per-task test AUROC on EHR Event Modelling: agents vs. the CLMBR and count+GBM baselines.** Bars are each agent's mean test AUROC across the six new-onset prediction targets. The solid black mark is EHRSHOT's published CLMBR model's performance on this testset ; the dashed magenta mark is the count+GBM baseline (gradient boosting over hand-engineered count features), which is the pass bar for the task. Both baselines are scored on the same first-prediction-time test cohort as the agents. Despite using only basic feature engineering, the best agent matches or exceeds the deep learning-based CLMBR baseline on every task (Both baseline numbers differ from those reported in the EHRSHOT paper [33] because we evaluate on a subset test cohort using strictly the first observation per patient to prevent leakage.)

E HealthAgentBench Task Categories

E.1 EHR Format Conversion: EHR ETL pipeline customization

Tags.

- *Modality:* EHR (MEDS standard target).
- *Workflow stage:* Data Management.
- *Output shape:* Directory tree (MEDS cohort parquet shards).
- *Data access:* Public (MIMIC-IV demo).
- *Expert time-to-solve:* 1–4 h
- *Number of Tasks* 1

Motivation. The task tests the agents for the ability to inspect a real open-source ETL repository and produces a config that changes the shape of an output cohort. Its contribution is not the upstream ETL codebase itself but the conversion of a fragmented, implementation-heavy healthcare workflow into a standardized, verifiable agent task.

Task setup.

- *Inputs (mounted read+write):* unfamiliar upstream clone at `/workspace/MIMIC_IV_MEDS`; demo input at `/workspace/staged_demo/raw_input`; writable `/workspace/output/`.
- *Instruction (verbatim, given to the agent):*

  > *Use the codebase at `/workspace/MIMIC_IV_MEDS` to run its ETL pipeline on the demo input at `/workspace/staged_demo/raw_input`. Inspect the default extraction config `event_configs.yaml`, copy it to `custom_event_configs.yaml`, and edit the copy — leaving the default config unchanged and the repo's default wiring intact — then run the ETL with the new config for this run only. The customized cohort must no longer store `insurance`, `language`, `marital_status`, or `race` as fields on the admission event, but record each as its own admission-time event (prefixes `INSURANCE//`, `LANGUAGE//`, `MARITAL_STATUS//`, `RACE//`); and must use `OMR//` for outpatient OMR measurements, `HOSP_LAB//` for `hosp/labevents`, and `ICU_CHARTEVENT//` for `icu/chartevents`.*

- *Required actions:* (i) create a new config by copying the default; (ii) edit the copy to produce the required transforms (admission demographics as timestamped events; OMR / HOSP_LAB / ICU_CHARTEVENT code-prefix families); (iii) run the pipeline with `uv`; (iv) write the final cohort to `/workspace/output/MEDS_cohort`.
- *Hidden (verifier-only):* gold summary (`gold_demo_summary.json`) and reference custom config (`reference_custom_event_configs.yaml`) under `/tests/`.
- *Submission:* the `/workspace/output/MEDS_cohort/` directory tree.
- *Limits:* agent timeout 7200 s, verifier timeout 900 s, 4 CPUs, 8 GB RAM, 20 GB storage, internet allowed.

Data source.

- *Upstream:* `MIMIC_IV_MEDS` (Medical Event Data Standard) [23].
- *Cohort:* open MIMIC-IV demo (v2.2) [19], public.

Tasks in this Category.

- *Count:* 1 task.

Scoring.

- *Checks (in `verify_output.py`):* (a) default config is byte-identical to upstream; (b) custom config exists and parses; (c) required code-prefix families (`INSURANCE//`, `LANGUAGE//`, `MARITAL_STATUS//`, `RACE//`, `OMR//`, `HOSP_LAB//`, `ICU_CHARTEVENT//`) are present in the cohort metadata; (d) per-shard parquet row counts and canonical content hashes match the gold summary.
- *Reward (Success Criterion):* Completion of all checks (a)–(d) yields 1.0 (success); any failure on (a)–(d) yields 0.0 (failure).
- *Raw Per-trial Metrics:* none beyond the binary reward (the verifier emits only success/failure).
- *Hash canonicalisation:* rows are sorted on `(subject_id, time, code)` before hashing, so the check is robust to harmless writer-side reordering.

Anti-cheat & leak-proofing.

- *Default-config integrity:* check (a) prevents the agent from short-circuiting the task by patching the upstream's default wiring.
- *Credential surface:* none. The demo data is public.

Bootstrap & environment.

- *Compose pattern:* single-service Docker.
- *Build-time work:* Dockerfile clones upstream at pinned commit and runs `stage_demo_data.py` to download the demo dataset.
- *Critical path:* `uv sync` inside the agent's runtime; failure to install dependencies is a common error.

Baseline observations. 10 frontier models at xhigh effort, 3 attempts each. Nine of the ten score a perfect 3/3; only `claude-opus-4-6` fails (0/3, $0.88), consistently dropping a demographic field from the output cohort. Among the passers, cost spans ∼8× — from `gpt-5.4-mini` ($0.29) and `gpt-5.3-codex` ($0.37) up to `claude-opus-4-7` ($2.34) — so the cheapest reliable pass is far cheaper than the priciest at identical reliability.

Known failure modes.

- Editing the default config in place instead of copying it.
- Omitting one or more of the seven required code-prefix families.
- Pointing the ETL run back at the default config.

Reference solution. A human-authored reference custom config ships at `scripts/ehr_to_meds_etl/assets/reference_custom_event_configs.yaml`. It documents the intended seven-prefix layout and serves as the sanity-checking artifact for any change to the verifier's gold summary. The reference is mounted only to the verifier and is never agent-visible; agents that produce semantically-equivalent but textually-different configs are accepted by the canonical content-hash check in `verify_output.py`.

E.2 X-ray Report Correction: chest X-ray report correction

Tags.

- *Modality:* Longitudinal Imaging (chest X-ray) + text.
- *Workflow stage:* Diagnosis.
- *Output shape:* Free-text radiology section (FINDINGS).
- *Data access:* PhysioNet credentialed (MIMIC-CXR).
- *Expert time-to-solve:* 15–60 min
- *Number of Tasks* 10

Motivation. First multimodal, edit-correction benchmark in HealthAgentBench. The agent reviews a counterfactual draft FINDINGS for a target chest X-ray study against the patient's longitudinal imaging history and produces a corrected FINDINGS that resolves the clinically-significant errors the draft introduces. Success requires repeated tool calls to inspect images, integration of visual observation with textual prior context, and structured-output discipline. HealthAgentBench's contribution is the conversion of a real-world, multimodal clinical workflow into a verifiable agent task within the data environment of the patients' longitudinal multimodal contexts.

Task setup.

- *Inputs (mounted at `/data/patient/` with opaque IDs, `study_NN_<timestamp>/view_NN.jpg`):* the patient's prior studies (JPGs + free-text reports + timestamps); the target study's non-generative sections (EXAMINATION, INDICATION, HISTORY, TECHNIQUE, COMPARISON); and the counterfactual draft FINDINGS (The draft is a perturbed version of the gold FINDINGS that introduces contradictions by swapping the presence/absence of key findings, the location of findings, or the severity of findings, or the temporal changes without adding any new findings that the gold does not mention. The agent must correct the draft by removing the introduced errors without adding any new findings.)
- *Instruction (verbatim, given to the agent):*

> *A draft* `FINDINGS` *section for the patient's most recent chest X-ray, written by a junior radiologist, is already populated in the target study's* `report.txt`*; review and correct it. You may edit existing sentences but may* not *add statements about findings the draft did not already mention, and you submit only the corrected* `FINDINGS` *(no* Impression*). Use the chest-X-ray images and the prior studies' reports to determine the correct findings. All data under* `/data/patient/` *belongs to one patient; each subfolder is one study, folders sort chronologically, and the highest-numbered* `study_NN` *is the target. Set* `final_answer` *in* `/workspace/submission.json` *to the corrected report, starting with a literal* `FINDINGS:` *header. You have up to 1 hour.*

- *Required action:* detect and correct the clinically-significant errors in the draft (editing existing sentences only, adding no new findings).
- *Output:* JSON at `/workspace/submission.json` with a `final_answer` string that begins with a literal `FINDINGS:` header followed by the corrected Findings body (no Impression).
- *Hidden (verifier-only):* gold Findings at `/tests/target_report.txt`; the corpus name and MIMIC subject IDs do not appear in any agent-visible path.
- *Limits:* agent timeout 3600 s, verifier timeout 900 s, 2 CPUs, 4 GB RAM, 10 GB storage, internet allowed.

Data source.

- *Corpus:* MIMIC-CXR v2.1.0 [18] + MIMIC-CXR-JPG v2.1.0 [17]; verifier uses CheXprompt [36].
- *Access:* PhysioNet credentialed.
- *Version pin:* MIMIC-CXR v2.1.0 PhysioNet release.

Tasks in this Category.

- *Count:* 10 cases (`case_01..case_10`).
- *Selection criteria:* the 10 cases were manually reviewed so each gold FINDINGS is clinically accurate (zero clinically-significant errors per expert annotation); cases whose gold itself had factual issues (wrong lateralization/severity, missed hardware, hallucinated priors) were filtered out. The set deliberately mixes longitudinal patients with $\geq$1 prior study (cases 01–05, 07, 09, 10) and single-study patients with no prior (cases 06, 08). Each case's draft FINDINGS is then synthesized at bootstrap by applying one or more documented swap principles (lateralization, severity, comparison-word flip, no-prior/no-change, count, location, negation) to the gold, introducing at least one clinically-significant error the agent must correct.

Scoring.

- *Per-trial:* CheXprompt LLM-as-a-judge 5-vote majority on the submitted Findings vs. held-out gold.
- *Reward (Success Criterion):* 1 iff $\geq 3/5$ votes report zero clinically-significant errors.
- *Default judge:* `gpt-5.4`.

- *Raw Per-trial Metrics:* mean clinically-significant errors.
- *Gold Label Review:* each case's gold FINDINGS were hand-reviewed against the paired chest X-ray and the patient's prior imaging history to ensure they are factually supported and free of clinically-significant errors, so the gold is a reliable target for correction.

Anti-cheat & leak-proofing.

- *Two-service split:* `bootstrap` holds PhysioNet credentials (`PN_USER`, `PN_PASS`); the `main` agent service has none. Therefore, the agent cannot simply download the gold FINDINGS from PhysioNet.
- *ID opacity:* the corpus name, MIMIC subject IDs, and study identifiers never appear in agent-visible filenames or paths.
- gold target FINDINGS are never agent-visible; they live only in the verifier.

Bootstrap & environment.

- *Compose pattern:* two-service. `bootstrap` fetches the credentialed PhysioNet release, extracts each case's gold FINDINGS to `/tests/target_report.txt`, enumerates and stages priors into `/data/patient/`, then terminates.
- *Gating:* `main` is gated on bootstrap's clean exit via docker-compose's `service_completed_successfully` condition.

Baseline observations. 10-row table, all at xhigh, 10 cases × 3 attempts = 30 trials per model. Codex `gpt-5.4` leads with a 0.400 mean success rate (12/30 successes, $0.43/trial), and also posts the fewest mean clinically-significant errors (1.46/case). Codex `gpt-5.5` (0.333, $0.90) is more expensive without a quality gain. Copilot CLI's `gpt-5.5` is the strongest non-Codex agent (0.267, 8/30). Three models tie for worst at 0.100 sucess rate (3/30 successes each): `gpt-5.4-mini`, `claude-opus-4-6`, and `claude-sonnet-4-6`. The split is driven by model family more than by harness: the four non-mini GPT-5 models (Codex `gpt-5.4`/`gpt-5.5`/`gpt-5.3` and Copilot `gpt-5.5`) take the top success rates and the four lowest mean clinically-significant-error counts (1.46–1.71/case), whereas every Claude-family agent (Claude Code opus/sonnet and Copilot `opus-4.8`) trails on both (2.55–4.80 errors/case).

Known failure modes.

- Agent introduces a new hallucinated finding while removing one of the draft's errors.
- Agent leaves a draft error in place because it disagrees with a prior study's findings.
- Submitted JSON does not parse, or `final_answer` is missing the required `FINDINGS:` header.

E.3 Clinical Trial Matching: patient–trial eligibility matching

Tags.

- *Modality:* Text (admission note + trial XMLs).
- *Workflow stage:* Treatment Planning.

- *Output shape:* Eligible NCT-ID set, confidence-ordered (plain text).
- *Data access:* Public (TREC-CT 2021).
- *Expert time-to-solve:* >8 h
- *Number of Tasks* 9

Motivation. A clinical-trial recruitment / eligibility-matching benchmark category in HealthAgentBench. This task category requires the agent to read a free-text patient admission note and a set of candidate clinical-trial XMLs, then identify all trials for which the patient is eligible. The agent must ensure that every inclusion criterion is met and no exclusion criteria are violated. The output is a list of eligible trial identifiers (NCT-IDs) ordered by confidence. This task tests the agent's ability to understand complex medical eligibility criteria, extract relevant information from unstructured text, and make accurate eligibility determinations based on structured trial data. HealthAgentBench's contribution is clinical matching task into an end-to-end workflow and provides the environment and evaluation framework to verify the agent's eligibility determinations against physician-judged gold standards.

Task setup.

- *Inputs (mounted at `/workspace/data/`, bind-mounted from a host cache):* the patient admission note at `/workspace/data/topic.txt` and the topic's judged-pool corpus of ∼300–450 ClinicalTrials.gov XMLs (standard schema) at `/workspace/data/trials/<NCT_ID>.xml`.
- *Instruction (verbatim, given to the agent):*

  > *You are given a single patient's free-text admission note and a directory of candidate clinical-trial documents. Identify all trials the patient is eligible for—i.e. relevant to the patient, meeting every inclusion criterion and none of the exclusion criteria. The patient note is at `/workspace/data/topic.txt` and one trial per candidate at `/workspace/data/trials/<NCT_ID>.xml` (standard ClinicalTrials.gov schema). Write `/workspace/submission/eligible_trials.txt` with one NCT identifier per line—all eligible trials and none the patient is excluded from—in descending order of your confidence. You have up to 1 hour; do not search the internet for benchmark answers.*

- *Required action:* identify every trial the patient is eligible for, ordered by confidence.
- *Output:* plain-text NCT-ID list (one per line, most-confident first) at `/workspace/submission/eligible_trials.txt`.
- *Tools:* standard Python environment; XML parsing left to the agent.
- *Limits:* agent timeout 3600 s, verifier timeout 300 s, 2 CPUs, 2 GB RAM, 4 GB storage, internet allowed.

Data source.

- *Corpus:* TREC Clinical Trials 2021 track [32].
- *Access:* public.
- *Pin:* 75 topics; 35 832 physician-judged qrels; 400k-trial ClinicalTrials.gov snapshot dated 27 April 2021. See `design/related_work/clinical_trial_matching.md`.

Tasks in this Category.

- *Count:* 9 tasks selected from TREC 2021.
- *Selection criteria:* The tasks are selected to cover patient profiles stratified across clinical specialties (ages 15–70, both sexes).

Scoring.

- *Raw Per-trial Metrics:* mean `recall_top_50` (recall from the agent's top-50 most-confident picks).
- *Reward (Success Criterion):* a trial passes (reward 1) iff recall over the agent's 50 most-confident picks equals 1.0 (every eligible trial is recovered within the top 50), else 0. We use recall rather than F1 or precision for determining the success criterion because in real-world trial recruitment, missing an eligible trial is a more critical failure than including some ineligible ones. To guard against the agent flooding the recall score by including a long tail of low-confidence picks, we impose the top-50 cutoff: the agent must rank all eligible trials within its 50 most-confident picks from around 400 trial pool to pass. The top-50 cutoff reflects the practical reality that trial recruiters typically only have the bandwidth to review a limited shortlist of candidates, and this will prefer agents who rank eligible trials higher rather than burying them in a long tail of low-confidence picks.
- *Gold Eligible Trials* are judged by TREC's physician annotators based on the trial's XML and the topic's admission note; To ensure no ambiguity of the eligible trials, we manually review the judged eligible trials again and remove trials that contain ambiguous and unverifiable inclusion criteria (e.g. "must be able to come to our hospital"). Each task contains 3–9 gold eligible trials (43 in total across the nine tasks).

Anti-cheat & leak-proofing.

- The TREC corpus is public; we disable web browser and remove any corpus metadata from agent-visible paths to prevent shortcutting.
- Gold eligible trials are never agent-visible; they live only in the verifier.

Bootstrap & environment.

- *Compose pattern:* per-topic XML download uses HTTP range requests into `scripts/clinical_trial_matching/assets/raw_cache/` (`.gitignored`).
- *Concurrency control:* a global host-side `flock` plus a bootstrap-sentinel handshake serializes the fetch across concurrent containers so multiple agents on the same host do not duplicate downloads or race on the cache.

Baseline observations. 10 frontier models at xhigh effort, 9 tasks × 3 attempts = 27 trials per model. Four agents separate from the field: Copilot CLI's `opus-4.8` leads at a 0.667 mean success rate (18/27), ahead of `gpt-5.5` (0.519, 14/27), `claude-opus-4-8` (0.481, 13/27), and Copilot's `gpt-5.5` (0.444, 12/27); every other agent scores at or below 0.259. The same four top the soft `recall_top_50` metric (all between 0.82 and 0.93).

Known failure modes.

- Agent misreads exclusion criteria as inclusion-style language.
- Agent triages with title and summary but misses eligibility details in the full XML.

E.4 CT Abnormality Classification: chest CT interpretation

Tags.

- *Modality:* 3-D imaging (NIfTI volume).
- *Workflow stage:* Diagnosis.
- *Output shape:* Per-volume binary label vector.
- *Data access:* HuggingFace gated under OpenRAIL (CT-RATE).
- *Expert time-to-solve:* 1–4 h
- *Number of Tasks* 10

Motivation. 3-D imaging benchmark in HealthAgentBench. Each task in this category require the agent to inspect a non-contrast chest CT volume and classify it for the presence or absence of a set of clinical findings. The agent must read the volume, interpret the imaging data, and produce a binary label for each requested finding. The tasks test the agent's ability to process volumetric medical imaging data, recognize relevant anatomical structures and pathologies, and make accurate diagnostic classifications based on visual evidence. HealthAgentBench's contribution is the conversion of a real-world, 3-D imaging clinical workflow into a verifiable agent task within the data environment of the patients' imaging context. The agent environment provides freedom for agents to explore data and propose strategies (eg. with internet to download libraries etc.).

Task setup.

- *Inputs (mounted at `/workspace/data/`):* one non-contrast chest CT (NIfTI, 200–700 slices, Hounsfield units) at `/workspace/data/scan.nii.gz`, and a per-volume label list (one label per line, typically 4–12 categories) at `/workspace/data/labels.txt`.
- *Instruction (verbatim, given to the agent):*

  > *You are working inside an environment that contains a single non-contrast chest-CT volume and a list of clinical findings to evaluate. For each listed finding, decide whether it is present or absent in the scan. Write a plain text file at `/workspace/submission/predictions.txt` with one `<label>: yes` or `<label>: no` line per requested label, using the exact name from `labels.txt` (case-insensitive); order does not matter. Solve using only the volume on disk; do not look up the dataset's published labels or report. The reward is binary — every label must be correct. You have up to 1 hour.*

- *Required action:* inspect the 3-D volume (the canonical path is `nibabel` → PNG slices → multimodal view) and classify each requested label.
- *Output:* text file at `/workspace/submission/predictions.txt`, one `<label>: yes/no` per line; order is irrelevant (the verifier matches by label name).

- *Tools:* standard Python environment with `nibabel`, `pillow`, `huggingface_hub`.
- *Hidden (verifier-only):* gold labels at `tests/gold.json` (gitignored, derived at bootstrap).
- *Limits:* agent timeout 3600 s, verifier timeout 300 s, 2 CPUs, 8 GB RAM, 8 GB storage, internet allowed.

Data source.

- *Corpus:* CT-RATE [11] — 25 692 non-contrast 3-D chest-CT volumes paired with radiology reports; validation split of 3 038 volumes.
- *Access:* HuggingFace gated under OpenRAIL; `HF_TOKEN` required at bootstrap.
- *Version pin:* CT-RATE validation split as of the dated snapshot in `design/related_work/ct_abnormality.md`.

Trial construction.

- *Count:* 10. Each task containes a volume from CT-RATE's validation split.
- *Selection:* The 10 tasks are selected to cover disease categories with a range from single disease to multiple diseases.

Scoring.

- *Reward (Success Criterion):* all disease labels are predicted correct $\rightarrow$ 1.0, any error $\rightarrow$ 0.0.
- *Gold Label Review:* each volume's gold labels were hand-reviewed against the paired radiology report to ensure every positive and negative label has a verbatim evidence phrase.
- *Raw Per-trial Metrics:* mean accuracy.

Anti-cheat & leak-proofing.

- *Token isolation:* `bootstrap` holds the `HF_TOKEN`; `main` has neither token nor report access. Therefore, the agent cannot simply download the gold labels from HuggingFace.
- *Gold isolation:* `tests/gold.json` lives only inside the verifier volume.

Bootstrap & environment.

- *Compose pattern:* two-service. `main` depends on `bootstrap`'s successful completion.
- *Build-time data:* bootstrap downloads 3.5 GB of NIfTI volumes (100–350 MB each) lazily into a shared host cache so re-runs on the same host avoid re-downloading.

Baseline observations. 10-row table, all at xhigh, 10 volumes $\times$ 3 attempts = 30 trials per model. Codex `gpt-5.5` attains the best success rate (10/30, 0.333). The other codex models follow: `gpt-5.4` at 9/30 (0.300) and `gpt-5.4-mini` at 7/30 (0.233), tied with Copilot's `gpt-5.5` (7/30). They are ahead of the Claude-family agents (`claude-opus-4-8` and Copilot `opus-4.8` both 5/30, `claude-opus-4-7` 4/30).

Known failure modes.

- *Under-calling.* The agent renders appropriate lung and mediastinal windows but, not seeing florid disease, defaults subtle findings (e.g. paraseptal emphysema, small nodules) to "absent" — so real findings are missed.
- *Over-calling.* Conversely — most often in the weaker models — the agent flags findings that are not present, adding false positives that likewise break the exact-match reward (e.g. a Sonnet-4.6 trial that caught the one real finding, lung opacity, but additionally marked cardiomegaly, pericardial effusion, and lymphadenopathy present, scoring 0.25 accuracy and reward 0). Over-calling is roughly as common as under-calling across the sweep: 85 failed trials are pure false-positive errors.

E.5 Pathology Tumor Area Selection: pathology whole-slide reasoning

Tags.

- *Modality:* Pathology whole-slide image (gigapixel).
- *Workflow stage:* Research/Diagnosis.
- *Output shape:* Tumor-tile coordinate set (JSON).
- *Data access:* Public (CAMELYON16).
- *Expert time-to-solve:* 1–4 h
- *Number of Tasks* 10

Motivation. Delineating the tumor area from surrounding tissue is a fundamental first step in numerous pathology workflows, and it underpins a wide range of downstream research and clinical applications biomarker quantification, grading, and the extraction of tumor-specific morphological features. The task is challenging for several reasons. First, whole-slide images are gigapixel in scale, on average around 100,000 × 100,000 pixels, which makes exhaustive inspection infeasible and forces multi-resolution reasoning: candidate regions are located at low magnification and confirmed at high magnification. Second, tumor is often sparse, or present only as small foci (e.g., micro-metastases or isolated tumor-cell clusters) that are easily missed. Third, the boundary between tumor and normal tissue is frequently ambiguous on hematoxylin and eosin (H&E) staining alone, such that even expert annotations are typically aided by immunohistochemical restaining. These properties make tumor delineation a non-trivial, decision-intensive task that requires actively navigating the slide, selectively inspecting local regions, and integrating evidence under a finite inspection budget. The HealthAgentBench contribution is turning CAMELYON16 into a reproducible Harbor-first agentic task with one task per slide, hidden gold segmentation, and pooled tile-level aggregate metrics. We do not require dense pixel-level segmentation; instead, the agent selects the grid tiles it believes contain tumor which is enough in many downstream research tasks. Note that CAMELYON16 comprises lymph-node whole-slide images from breast-cancer patients, and its reference masks delineate *metastatic* breast carcinoma deposited in the node rather than a primary tumor. Because a metastasis is itself neoplastic tumor tissue, we treat these annotated regions as the tumor area to be selected, and "tumor-area selection" here corresponds to localizing nodal tumor (metastatic) deposits.

Task setup.

- *Inputs (mounted into the environment):* one whole-slide pathology image at `/data/slide/current/slide.*` (pyramidal `.tif`, read via OpenSlide/tifffile) and the analysis-grid metadata in the public task row `/workspace/benchmark_tasks.json`.
- *Instruction (verbatim, given to the agent):*

  > *You are working in a pathology task environment. The current whole-slide image is at `/data/slide/current/slide.*`, a public task row at `/workspace/benchmark_tasks.json`, and an editable submission at `/workspace/submission.json`. The analysis grid uses `256x256` tiles at downsample 16, so each grid tile spans `4096x4096` full-resolution (level-0) pixels. Identify all and only the tiles that contain tumor and populate `predicted_tumor_tiles` with objects of the form `{"x": <int>, "y": <int>}` using integer grid coordinates; treat non-tissue tiles as non-tumor. You have a budget of 1.5 hours; do not look up solutions online and do not train or fine-tune models.*

- *Required action:* predict the complete set of tumor tiles on the fixed 256×256 grid at down-sample 16; a tile is "tumor" if $\geq 20\%$ of its area is tumor.
- *Output:* JSON at `/workspace/submission.json` with `predicted_tumor_tiles`, a list of `{x, y}` grid-coordinate objects. (A `contains_tumor` boolean is also recorded but is not scored here.)
- *Tools:* by default—the configuration used for our baselines—the agent reads the slide itself via OpenSlide/tifffile with no helper scripts;.
- *Hidden (verifier-only):* gold tumor masks under `/tests/` (never mounted into `main`).
- *Limits:* agent timeout 5400 s, verifier timeout 3600 s, 2 CPUs, 8 GB RAM, 20 GB storage, internet allowed.

Data source.

- *Corpus:* CAMELYON16 (AWS S3 public mirror at `s3://camelyon-dataset/CAMELYON16/`); slides are pyramidal `.tif` read via OpenSlide.
- *Access:* public, no credentials.

Tasks in this Category.

- *Count:* 10 tasks with each contains a CAMELYON16 tumor slide
- *Selection:* The slides inherit a range of tumor sizes from CAMELYON16 (gold regions span ≈20–160 tiles) and varied metastasis morphology; smaller deposits are missed even by strong agents.

Scoring.

- *Reward (Success Criterion):* Calculate tile-level F1 of the predicted tile set against the hidden gold mask; a trial passes (reward 1.0) iff tile-F1 ≥ 0.90, else 0.0. We choose 0.9 F1 rather than 1.0 F1 to allow for some small differences due to ambiguity in the tumor boundary.
- *Raw Per-trial Metrics:* tile-level F1.

Anti-cheat & leak-proofing.

- *Gold isolation:* gold tumor masks live under `/tests/` (verifier-only).
- *Credential surface:* none (datasets are public).

Bootstrap & environment.

- *Compose pattern:* two-service (one-shot `bootstrap` then `main`), with lazy public-data download into `scripts/tumor_area_selection_pathology/assets/raw_cache/` (gitignored shared host cache).
- *Cache sharing:* multiple containers on the same host share the cache so the gigapixel slides download once per host, not once per trial.

Baseline observations. 10 frontier models at xhigh effort, 10 tasks $\times$ 3 attempts = 30 trials per model. `gpt-5.5` leads, passing 12/30 (mean success rate 0.400) at the highest tile-F1 (0.845), and is the only agent above 0.30. `claude-opus-4-8` follows at 0.200 (6/30); a cluster of three sits at 0.167 (5/30): `claude-opus-4-6`, `gpt-5.3-codex`, and Copilot's `opus-4.8`, and the rest trail from there (Copilot's `gpt-5.5` 4/30; `claude-opus-4-7` and `claude-sonnet-4-6` 3/30; `gpt-5.4` 2/30; `gpt-5.4-mini` 0/30). The pass gate (tile-F1 $\geq$ 0.90) is demanding: `gpt-5.4-mini` attains 0.967 tile recall but only 0.320 precision (it over-selects tiles), so its tile-F1 (0.481) never clears the bar.

Known failure modes.

- Agent skims via thumbnail and misses small tumor foci.
- Agent over-segments tissue artifacts as tumor.
- Agent confuses adjacent benign tissue (e.g. inflammation) with tumor.

E.6 `ehrshot`: longitudinal clinical event prediction

Tags.

- *Modality:* Longitudinal EHR timeline (tabular events).
- *Workflow stage:* Research
- *Output shape:* Per-patient probability (CSV).
- *Data access:* Redivis-gated (Stanford STARR via EHRSHOT).
- *Expert time-to-solve:* 4–8 h
- *Number of Tasks:* 6

Motivation. This task category challenges the agents with longitudinal clinical event prediction in HealthAgentBench. The agent must learn a prediction rule from labelled training data and apply it to an unlabelled held-out test set. The HealthAgentBench contribution is packaging the EHRSHOT raw-timeline benchmark as a reproducible Harbor-first task family with one-click container-side data download, hidden test labels enforced by docker-compose volume topology, mechanical leak-proofing on the event slice, and pass-the-baseline success/failure gates.

Task setup.

- *Inputs (mounted at `/data/`):* training labels (`train_labels.csv`), validation labels (`val_labels.csv`), an unlabelled test set (`test_examples.csv`, no labels), and a per-task leak-proof slice of the longitudinal event log (`events.csv`; for test patients only events with start $< T_{\text{first}}$ are kept). Notice that the test set differs from the original EHRSHOT test set: we take the original test set and only retain the first prediction-time event slice for each test patient, so the agent must predict from partial timelines without any future information.
- *Instruction (verbatim; the hyperlipidemia target shown):*

  > *Predict whether the patient will have their first diagnosis of hyperlipidemia (code `SNOMED/55822004` and its ontology children) within the next year. Predictions are made at 11:59 PM on the day of discharge from an inpatient visit; discharges where the patient already has the diagnosis are ignored. Learn a strategy from the labelled `train` and `val` splits and apply it to a held-out `test` split given without labels, using only events with `start` before each prediction time. Write `/workspace/submission/predictions.csv` with columns `patient_id,prediction_time,probability` (continuous in* $[0, 1]$*). Scored by AUROC. You have 1 hour and internet access.*

- *Required action:* learn a prediction strategy from train + val, apply to the test set.
- *Output:* per-patient probabilities at `/workspace/submission/predictions.csv`, columns `patient_id,prediction_time,probability`; scored by AUROC.
- *Hidden (verifier-only):* test labels live on a `/tests/` mount that is *not* mounted into `main`; the Redivis API token never reaches `main`.
- *Limits:* agent timeout 3600 s (1-hour budget), verifier timeout 300 s, 16 CPUs, 64 GB RAM, 65 GB storage. Internet *enabled* inside main.

Data source.

- *Corpus:* EHRSHOT [33] on the Stanford STARR EHR cohort (6 739 patients, 41.6M observations).
- *Access:* Redivis-gated; access via `~/.redivis/api_token`.
- *Pin:* 4 GB Redivis bundle per task slice. See `design/related_work/ehrshot_2307.02028.md`.

Tasks in this Category.

- *Count:* 6 tasks
- *Selection Criteria* We select 6 tasks from EHRSHOT benchmark that focus on new-onset diagnosis predictions (hyperlipidemia, celiac disease, acute myocardial infarction, pancreatic cancer, systemic lupus (SLE), essential hypertension).

Scoring.

- *Reward (Success Criterion):* AUROC is computed on agent's submission against the held-out test labels. 1 iff AUROC $\geq$ the task's count+LightGBM baseline (per-task baseline JSON hardcoded in the generator), else 0.
- *Raw Per-trial Metrics:* test AUROC.

Anti-cheat & leak-proofing.

- *Event-slice integrity:* for test patients, events are kept only where `start < T_first` (the patient's first prediction time) and any future `end` value is blanked; train/val patients keep their full timelines.
- *Test-label isolation:* test labels are never committed to git; bootstrap writes them to a `/tests/` mount that `main` does not see.

Bootstrap & environment.

- *Compose pattern:* two-service. `bootstrap` reads `~/.redivis/api_token`, downloads the 4 GB bundle, slices the event log, partitions splits, writes test labels to `/tests/`, and terminates.
- *Resource profile:* 16 CPUs, 64 GB RAM, 65 GB storage — the most expensive task in the suite.

Baseline observations. 10 frontier agents at xhigh effort, 6 targets × 3 attempts = 18 trials per agent. `claude-opus-4-7` leads at a 0.778 mean pass rate (14/18), with Codex `gpt-5.5` and Copilot's `opus-4.8` next (both 0.722, 13/18) and `claude-opus-4-8` and Copilot's `gpt-5.5` just behind (both 0.667, 12/18); three more tie at 0.611 (11/18): `claude-sonnet-4-6`, `gpt-5.3-codex`, and `gpt-5.4`. `gpt-5.4-mini` trails at 0.389 (7/18), and the weakest, `claude-opus-4-6`, passes only 0.333 (6/18), incurring 7 agent timeouts at the longest wall-clock in the suite. Mean test AUROC is tightly bunched (0.72–0.77) across all agents, so the pass-the-baseline gate, rather than raw discrimination, drives the spread. Per-trial cost ranges from $0.80 (`gpt-5.3-codex`) to $3.32 (`gpt-5.5`).

Known failure modes.

- *Overfitting on rare outcomes.* With few positives (e.g. celiac disease has only ∼11 validation positives), the agent tunes a feature/ensemble blend that scores well on its own validation but collapses on the held-out test set — GPT-5.5's best celiac model fell from ∼0.74 validation AUROC to 0.30 on test, *below chance*.
- *Self-inflicted validation leakage.* The agent's own evaluation harness leaks (e.g. ensemble scores bleeding across folds), inflating its AUROC estimate and masking the generalization gap until the held-out test exposes it.
- *Resource exhaustion.* Weaker agents time out while streaming the multi-gigabyte event log (`events.csv` is ∼2.1 GB) and produce no submission.

E.7 EHR Data Quality Auditing: EHR data-quality auditing

Tags.

- *Modality:* Tabular EHR.
- *Workflow stage:* Data Management
- *Output shape:* Flagged-row CSV (`table,_row_id`).
- *Data access:* Public (MIMIC-IV demo + synthetic errors).

- *Expert time-to-solve:* >8 h
- *Number of Tasks:* 8

Motivation. This task category introduces a data-quality auditing benchmark in HealthAgentBench. The agent acts as an auditor of a structured EHR dataset, flagging rows that violate clinical plausibility. The HealthAgentBench contribution is a deterministic, reproducible synthetic-error benchmark with hidden labels, source-name obfuscation, and scoring that combines cluster-level recall with row-level precision.

Task setup.

- *Inputs (mounted at `/workspace/data/csv/`):* the corrupted EHR-demo subset as gzipped CSVs (`<table>.csv.gz`), spanning 8 hospital and ICU tables. The corruptions are generated by a deterministic synthetic-error recipe seeded at Docker build time including the following error types: impossible values (range-extreme, decimal-shift, unit-confusion, and unit-label mismatch), inconsistencies (conflicting/duplicate records both in-table and cross-table), and demographic contradictions (patient demographics that contradict other records).
- *Instruction (verbatim; the demographic-conflict task shown):*

  > *You are working inside a task environment that contains a copy of an EHR dataset under `/workspace/data/`. Do a data-quality check and flag data-entry errors belonging to certain error categories. Flag all errors in the category* demographic contradictions *— the patient's recorded demographic information contradicts other evidence about that patient. Submit a CSV at `/workspace/submission/flagged_rows.csv` with columns `table` and `_row_id`. You have up to 1 hour; do not look up solutions on the internet.*

- *Instruction disclosure:* the base instruction names the error family but not its sub-types; the `_clues` variant additionally discloses the sub-types and the tables to inspect.
- *Required action:* flag rows the agent believes are problematic.
- *Output:* CSV with the schema `table,_row_id` at `/workspace/submission/flagged_rows.csv`.
- *Limits:* agent timeout 3600 s, verifier timeout 600 s, 2 CPUs, 4 GB RAM, 10 GB storage, internet allowed.

Data source.

- *Corpus:* MIMIC-IV demo (v2.2) [16] with a deterministic synthetic-error injection recipe applied at Docker build time.
- *Pin:* the synthetic-error recipe is seeded; gold is derived alongside the corruption.

Tasks in this Category.

- *Count:* 8 tasks
- *Task Names*: four base subtasks plus an instruction-disclosure (`_clues`) variant of each. The four base subtasks are: `task_impossible_value`, `task_inconsistency`, `task_demographic_conflict`, and `task_combined`. The `task_combined` task mixes all three error families. The `_clues` variant of each subtask is identical in data, labels, and scoring but the instruction additionally names the injected sub-types and the tables to inspect.

Scoring.

- *Reward (Success Criterion)* a trial passes iff cluster-level recall is 1.0 (every injected error cluster has at least one flagged member). We choose recall as there could be already be errors in the real database. To guard against the agent flooding the submission with false positives, we also require that precision is at least 0.01;
- *Raw Per-trial Metrics:* mean cluster-level recall. A cluster is a set of rows that are all part of the single injected error (e.g., a set of conflicting records). The mean cluster-level recall is computed as the fraction of clusters for which at least one row was flagged by the agent.
- *Gold:* compared against a hidden gold set produced by the error-injection recipe. We manually review each error to ensure it is genuine error which should be flagged.

Anti-cheat & leak-proofing.

- *Source-name obfuscation:* agent-visible CSV filenames are renamed; the term "MIMIC" does not appear in any agent-visible path.
- *Instruction-level prohibition:* the agent is told not to fetch the upstream dataset; combined with source-name obfuscation, this is the live defense against pulling a clean copy. (A planned `/etc/hosts` egress block is not currently active, since `/etc/hosts` is read-only at image-build time.)

Bootstrap & environment.

- *Compose pattern:* single-service Docker.
- *Build-time work:* demo CSVs are downloaded and corrupted deterministically at build time using a seeded synthetic-error recipe; the gold set is computed alongside and held by the verifier.

Baseline observations. 10-row table, all at xhigh, 8 tasks × 3 attempts = 24 trials per model. `claude-opus-4-6` and Copilot's `gpt-5.5` lead, tied at a 0.417 mean success rate (10/24), with `claude-opus-4-7`, `claude-opus-4-8`, and Copilot's `opus-4.8` next at 0.333 (8/24) and Codex `gpt-5.5` at 0.250 (6/24); the remaining codex models and `claude-sonnet-4-6` sit at 0.125 or below. No agent clears 0.42, making this one of the hardest benchmarks in the suite. The pooled mean recall gives us a different angle: most agents recover the large majority of injected error clusters (the strongest exceed 0.85 mean recall), yet rarely catch *every* cluster in a task. Because the pass gate requires cluster recall = 1.0, these high-recall-but-incomplete runs still score 0.

Known failure modes.

- Agent under-flags: misses subtle conflicting/duplicate records.
- No agent over-flags to reach the precision floor of 0.01.

F Trajectory Analysis for Successful Trials

To qualitatively understand *how* the strongest agents solve HealthAgentBench tasks, we sampled, for every benchmark category, the best-performing agent's trajectories which capture every reasoning message, shell command, and observation). This appendix walks through one representative success trajectory per task category and summaries the strategy each agent used.

A behavioral signature recurs across all seven task categories: the strongest agents spend the majority of their steps *orienting and verifying*, enumerating the workspace, reading source/data, and checking their own intermediate output, rather than generating code or answers blindly. The per-category subsections below make this concrete.

F.1 EHR Format Conversion: minimal scoped config override

Best agent: Codex GPT-5.5 (success rate 100%; 59 steps, $1.01). The task asks the agent to make a MIMIC-IV→MEDS ETL emit a cohort whose admission demographics become separate admission-time events and whose measurement domains carry distinct code prefixes. The agent opens by stating its plan—"inspect the repo wiring and config first, then make the smallest scoped change"—and spends the first third of the run reading before any edit:

```
pwd && rg --files                                        # enumerate the repo
sed -n '1,240p' .../configs/event_configs.yaml           # read default config
find /workspace/staged_demo/raw_input -type f            # inspect staged input
cp .../event_configs.yaml .../custom_event_configs.yaml   # copy, don't edit
```

It then patches *only* the copied config and a CLI override ("the default YAML remains untouched and still remains the fallback"), runs the ETL, and validates the produced parquet contents—not just the exit code—before a final `git status` sweep to leave a minimal diff. This "orient → scope → execute → verify → clean" loop is the template the other tasks vary on.

F.2 Clinical Trial Matching: triage, then parallel eligibility adjudication

Best agent: Copilot CLI Opus 4.8 (success rate 67%); the sampled `task_19` trial retrieves 1.0 recall). Facing 301 candidate trial XMLs for one patient—a 65-year-old man with CAD and prior MI, recurrent sustained monomorphic VT, syncope, symptomatic bradycardia, and a resuscitated cardiac arrest, admitted for catheterization and an electrophysiology study—the Opus-4.8 orchestrator does not read every trial itself; it triages with code, then fans the eligibility decision out to parallel subagents:

1. *Scripted triage.* It writes a chain of inline Python passes over all 301 XMLs—parsing each trial's title, conditions, age/sex limits, and criteria; applying hard demographic filters (age 65, male); scoring relevance against cardiology keyword groups (VT/arrhythmia, ICD/SCD, syncope, CAD/MI, cath/PCI, HF, HTN, HLD); and *splitting* the surviving relevant subset into five disjoint batches.

2. *Parallel adjudication.* It spawns five background `general-purpose` subagents, one per batch, each prompted as a "clinical trial eligibility adjudicator" with a structured patient profile. Crucially, that profile spells out what the patient *does not* have— "no atrial fibrillation… does NOT currently have an ICD or pacemaker (only a loop recorder)… NOT nonischemic cardiomyopathy"—so each subagent applies the exclusion criteria consistently across its batch.

3. *Merge and re-verify.* It collects each subagent's verdicts, then re-reads the full eligibility text of the surviving and borderline trials itself (e.g. pulling DAVID II's criteria in full) to resolve edge cases, before writing `eligible_trials.txt` in descending confidence with a documented patient summary and validating that every NCT ID resolves to a real trial file.

This parallel-subagent orchestration is the opposite of the single-threaded Codex agent (the next-best, task success rate 52%), which reads far fewer trials but the *right* ones in one context. Opus-4.8 instead triages broadly with code, parallelizes the per-trial adjudication, then re-checks the borderline calls centrally. Because the pass gate rewards *recall* (every eligible trial recovered), the broad-then-verify fan-out pays off: it recovers all four gold-eligible trials (recall 1.0, a pass) at the cost of precision (0.40; six false positives among ten picks).

F.3 EHR Event Modelling: leakage-safe ML pipeline

Best agent: Claude Code Opus-4.7 (success rate 78%; the sampled trial passed with auroc above baseline). The agent must build a machine-learning pipeline over a multi-gigabyte event log and beat a baseline AUROC. Its summary captures the shape: "Loaded 29M events with pyarrow (∼8s) and built per-patient sorted timelines; extracted features for each (patient, prediction-time) pair" using only pre-cutoff events. Codex GPT-5.5 solves it the same way at comparable quality, narrating the key engineering decisions: it probes the installed stack ("XGBoost is installed, LightGBM. . . unusable because `libgomp` is missing"), uses Polars lazy joins to relate the 29.3M-row log to the prediction-row table by patient and pre-cutoff time ("42.2M pre-cutoff event-row pairs. . . in about 2.4 seconds"), enforces the anti-leakage cutoff *inside* the feature table to construct clean train data for generalization. Both agents win by treating data leakage as a structural invariant to construct train data rather than an afterthought.

F.4 CT Abnormality Classification: multi-window rendering and noise control

Best agent: Codex GPT-5.5 (success rate 33%; the sampled trial passed with 1.0 accuracy). Because the agent cannot see the 3-D volume directly, its entire strategy is to *manufacture the right views*: it reads voxel spacing, then renders cropped lung- and mediastinal-window contact sheets, axial/coronal/sagittal reformations, and—to fight non-contrast noise—"small z-slab mean images through the mediastinum," supplemented by HU measurements on suspected fluid pockets. It reasons per label (effusions, hiatal hernia, lymphadenopathy, lung opacity/consolidation), makes deliberately *conservative* binary calls when evidence is ambiguous, and finishes with a format check that every requested label appears exactly once with a `yes`/`no` value. The bottleneck is perceptual grounding, and the agent's edge comes from disciplined view generation rather than extra search.

F.5 Pathology Tumor Area Selection: multi-resolution tiling and morphology-driven calls

Best agent: Codex GPT-5.5 (success rate 40%; the sampled `slide_0001` trial passed with 0.967 tile f1), 163 true-positive tiles and no misses). The agent must mark which tiles of a gigapixel pathology whole-slide image contain tumor, scored on tile F1 and tumor coverage. As in CT Abnormality Classification, it cannot view the slide directly, so its whole strategy is to *manufacture aligned views at multiple magnifications* and then reason morphologically:

1. It reads the slide through the OpenSlide pyramid and renders views at several levels aligned to the scoring grid—a low-power (level-6) overview with the tile grid labelled, plus level-4 and

level-2 contact sheets per region—stating it will "generate downsampled views aligned to the 4096-pixel grid so the tumor tile calls can be made against the required coordinates."

2. It distinguishes tumor from confounders by histology rather than by a color threshold: "tumor versus lymphoid tissue has to be decided morphologically," identifying "the large left mass is carcinoma, while the upper trabeculated/dark areas are largely lymphoid or non-tumor," and computing per-tile tissue/stain fractions only to avoid over-calling low-tissue margins.

3. It then refines the boundary tile-by-tile—inspecting the ambiguous "transition row… where carcinoma along the lower edge" mixes into lymphoid tissue at full level-2 resolution—and runs an "outside pass" over high-tissue tiles it had not yet claimed to catch any overlooked carcinoma, overlaying the final selection on the overview before writing the tile coordinates to `submission.json`.

The trajectory mirrors CT Abnormality Classification almost exactly: the bottleneck is perceptual grounding, and the agent's edge comes from disciplined multi-resolution view generation and patient edge refinement rather than any learned classifier—no model is trained; every call is a visual judgement on a rendered tile. Figure 13 shows the outcome of this strategy on one slide, where Codex GPT-5.5 recovers the whole tumor region with only two spurious tiles (tile-F1 0.967).

F.6 X-ray Report Correction: read priors, let the image decide

Best agent: Codex GPT-5.4 (task success rate 40%). We sampled one success trajectory with just 24 steps, $0.314. This is the most economical passing trajectory in the suite. The agent identifies the target study, reads the draft `FINDINGS` and the prior reports to enumerate the claims at issue, then *consults the chest radiograph to resolve the specific contradictions* rather than rewriting wholesale: "The image confirms severe hyperinflation / emphysematous change without acute focal airspace disease, pleural effusion, or pneumothorax." It edits only the contradicted sentences, then does a JSON read-back. The discipline of grounding each correction in the image (and touching nothing else) is exactly what the edit-only scoring rewards.

F.7 EHR Data Quality Auditing: scripted detection with iterative recall recovery

Best agent: Claude Code Opus-4.6 (success rate 42%; the sampled `task_demographic_conflict` trial passed with full recall of errors). The agent inspects the eight gzipped tables, then writes and iteratively refines a Python detector for demographic contradictions:

```
for f in /workspace/data/csv/*.csv.gz; do zcat "$f" | head -3; done   # schema
python analyze.py        # first detector pass
python analyze2.py       # deeper: gender-specific labs, ref ranges, heights
python analyze3.py       # final comprehensive sweep
```

Its strength is *recall recovery through iteration*: after an initial pass it deliberately "digs deeper into gender-specific lab tests… and other demographic contradictions," catches missed Hemoglobin rows for two patients, discovers an additional gender-mismatched patient on a later sweep, and only then rebuilds the submission—verifying every flagged row id exists in the source. This pays off on a single error family but, as Section 4 and Appendix D show, no agent sustains it across the combined task.

F.8 Common themes

Two qualitative patterns hold across all seven tasks. First, *verification is a first-class activity*: successful agents read source and produced artifacts, re-check borderline decisions, and sweep their diffs, spending far more steps on checking than on writing. Second, *the strategy is shaped to the task's true bottleneck*—minimal config overrides for ETL, scripted triage followed by criterion-level reasoning for trial retrieval, leakage-safe joins for prediction to construct train and validation set, multi-resolution view generation for the imaging tasks (CT windowing and whole-slide tiling), image-grounded edits for report correction, and iterative recall recovery for data-quality auditing.

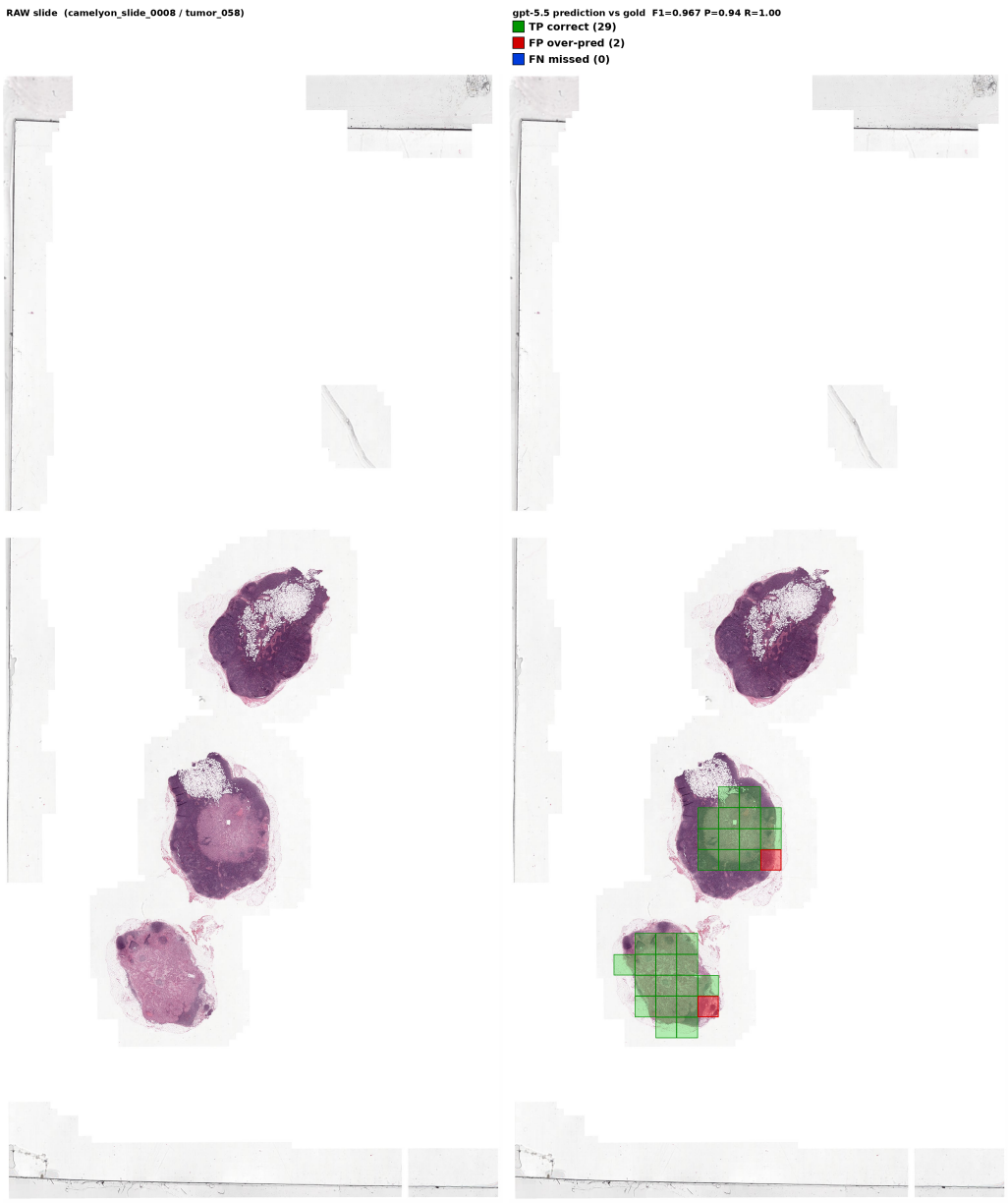


Figure 13: **A successful Pathology Tumor Area Selection trial (Codex GPT-5.5, slide `0008`).** *Left:* the raw whole-slide image (CAMELYON16 `tumor_058`), three lymph-node fragments at low power. *Right:* the agent's predicted tumor tiles overlaid on the gold mask — green are correct (29 true positives), red are spurious (2 false positives), and there are no missed tiles (0 false negatives). Codex GPT-5.5 localises the metastasis to the lower fragments and recovers the entire tumor region (recall 1.00) with only two over-predicted tiles, giving tile-F1 0.967 (precision 0.94), well above the 0.90 pass threshold. The benign upper fragment is correctly left unmarked.