READY or Not: *Reliable Enterprise Agent Deployment*

Veronica Chatrath[*1], **Bryan Zhu**[*1], **Jingxuan Fan**[*1], **George Pu**[1], **Soham Dinesh Tiwari**[1], **Soham Dan**[1], **Ryan Young**[1], **Yuan (Christy) Li**[1], **Yuang Yao**[1], **Apaar Shanker**[1], **Minglai Yang**[1], **Daniel Yue Zhang**[1], **Yunzhong He**[1], **Ying Liu**[1], **Chenguang Wang**[1,2], **Zhijun Yin**[1,3], **Yuan (Emily) Xue**[1]

[1]Scale AI, [2]University of California, Santa Cruz, [3]Vanderbilt University Medical Center, [*]Co-first authors.

veronica.chatrath@scale.com, yuan.xue@scale.com | https://scale.com/research

Abstract

An AI agent can perform well on benchmarks and still be unsuitable for deployment. Existing AI-agent benchmarks primarily measure whether an agent can complete realistic professional work, whereas enterprise deployment requires asking a different question: whether an agent can meet a required reliability level, under an acceptable level of human oversight, and at a tolerable cost. We introduce *Reliable Enterprise Agent Deployment* (READY), an evaluation framework for qualifying AI agents for deployment on concrete enterprise workflows. READY preserves each workflow's own definition of successful execution while applying a common deployment-qualification procedure. Given an agent, a workflow, and a class of candidate oversight policies, READY measures the reliability and operating cost of the human-AI system, selects the minimum-cost policy that satisfies a specified reliability target, and statistically qualifies the selected policy on held-out cases. The *deployment profile* characterizes the operating point supported by the evidence, including reliability, human-oversight burden, and cost. READY is implemented as an open testbed that decouples workflow specification, execution, evaluation, and deployment qualification, and runs on existing agent-evaluation infrastructure. In an end-to-end clinical-audit case study spanning 16 agent systems and 750 cases, READY reveals deployment differences hidden by autonomous benchmark performance: two systems separated by only 0.3 percentage points in autonomous accuracy (72.8% vs. 72.5%) require 39.2% versus 29.6% human review, respectively, to qualify at the same 76% reliability target under the evaluated oversight policy. Systems with nearly identical autonomous performance can support substantially different reliability–oversight tradeoffs. Accordingly, READY shifts enterprise agent evaluation from asking only *how well can the agent perform the work?* to asking *under what conditions, and at what cost, can it be reliably deployed?* By making those conditions explicit and statistically testable, READY provides a practical basis for comparing agent systems, setting oversight requirements, and making evidence-based deployment decisions.

1 Introduction

Enterprises are starting to hand real professional work to AI agents that reason over domain knowledge, use tools, and carry out multi-step tasks. A growing number of benchmarks evaluate these capabilities on professional workflows [1–3], scoring whether an agent can complete the task and how good the result is. These benchmarks mainly answer an important question about *capability*. However, whether to deploy an agent is a different question. An organization rarely needs an agent to work unattended. Rather, it deploys a system in which some work may be handled autonomously while other work is reviewed, corrected, approved, or taken over by humans. A more relevant question is not how often the agent succeeds on its own, but whether the human–AI system around it can achieve the reliability a particular workflow requires, how much human oversight that takes, and what the operating policy costs. An agent that is right on 80% of cases is not, by that fact alone, ready or unready to deploy. What matters is whether the wrong 20% can be identified and sent to a human, and how much that review adds to the cost of running the system.

We introduce *Reliable Enterprise Agent Deployment* (READY), a framework for turning workflow-level evaluation evidence into a deployment qualification. READY takes as input an agent, an enterprise workflow, a set of representative cases, and a class of candidate oversight policies. Each workflow retains its own definition of successful execution, including outcome correctness, process requirements, evidence grounding, policy adherence, and other requirements appropriate to that form of work. READY then evaluates the human–AI system under candidate oversight policies and asks which policy can satisfy a specified reliability target at minimum operating cost.

Professional-work benchmarks ask how well an agent can do the work autonomously.
READY asks under what conditions, and at what cost, an organization can reliably deploy it.

At its core, READY casts deployment qualification as a constrained optimization problem over a class of oversight policies. For each workflow, the evaluator maps a multidimensional set of deployment-relevant criteria into a measure of successful execution, while the oversight policy specifies the points at which human intervention may occur. Agent executions on representative cases provide the empirical evidence for estimating the reliability and operating cost induced by each candidate policy. READY then searches the policy class for the lowest-cost operating policy that satisfies a specified reliability target and any additional deployment constraints. The selected policy is frozen and statistically qualified on held-out cases not used during policy selection. The resulting *deployment profile* characterizes the reliability, oversight burden, cost, and workflow-specific risk of the qualified human–AI configuration.

We use a retrospective clinical-audit workflow [4] as a running case study to illustrate READY end to end. In this setting, an agent receives a longitudinal patient record and a clinical audit question, such as whether an ICU patient developed acute kidney injury within 48 hours of a CT scan. It must identify the relevant evidence in the record, apply the governing clinical standard, and return a verdict together with a stated confidence. A capability benchmark stops at the verdict and scores it. However, deployment requires an additional decision: which completed audits can be accepted automatically and which should be escalated to a clinician. That oversight policy, rather than raw verdict accuracy alone, determines the reliability of the human–AI system, the review burden it imposes, and its operating cost.

Across 16 agent systems and 750 cases, the deployment profiles reveal differences that autonomous benchmark performance alone does not. For example, two systems separated by only 0.3 percentage points in autonomous accuracy (GPT-5.4's 72.8% versus Sonnet 5's 72.5%) require 39.2% versus 29.6%

human review, respectively, to qualify at the same 76% reliability target under the evaluated oversight policy. More generally, the reliability–oversight frontier shows that systems with similar autonomous performance can support substantially different operating points. On a conventional leaderboard, the two systems appear nearly identical, yet their deployment costs differ substantially.

What READY adds. READY complements rather than replaces existing agent evaluations. Capability and work-product benchmarks [1–3] measure the quality of agent execution but do not determine an operating point for a deployed human–AI system. The $pass^k$ metric of τ-bench [5] measures consistency across repeated attempts but does not incorporate human oversight or operating cost. Cost-aware routing and cascades [6, 7] optimize quality–cost tradeoffs by routing among models, while selective prediction and learning-to-defer study when a model should abstain or pass a case to a human [8]. READY brings these ideas into a workflow-level deployment setting: it evaluates the reliability and cost of the complete human–AI system, optimizes an oversight policy against an explicit reliability requirement, and statistically qualifies the selected operating point on held-out evidence. It reuses the evidence produced by existing workflow evaluations instead of displacing them.

An open, extensible testbed. To support heterogeneous enterprise workflows, READY separates workflow-specific definitions of successful work from a common deployment-qualification procedure. We implement READY as an open, extensible testbed built on existing agent-evaluation infrastructure [9]. Workflows may differ in their task populations, execution environments, evaluation criteria, and forms of oversight while sharing the same qualification abstraction. We provide a seed workflow to demonstrate complete end-to-end use of this interface, and new workflows, cases, environments, and evaluators can be contributed while retaining their own standards of correct work.

Our contributions are as follows.

1. **A deployment-centered evaluation objective.** We formulate agent deployment as evaluation of a human–AI system under an oversight policy, moving from the question of how well an agent performs autonomously to which policy can meet a workflow-specific reliability requirement and at what operating cost (Section 2).
2. **A method for optimizing and statistically qualifying oversight.** READY selects a minimum-cost oversight policy subject to a reliability target and qualifies the frozen policy on held-out data. The deployment profile characterizes the reliability–oversight–cost operating point supported by the evidence, not autonomous capability alone (Section 3).
3. **An open evaluation interface and end-to-end empirical instantiation.** We provide an extensible interface that separates workflow evaluation from deployment qualification and instantiate it on a retrospective clinical-audit workflow [4] across 16 agent systems and 750 cases, demonstrating that similar autonomous performance can correspond to substantially different reliability–oversight tradeoffs (Sections 4–5).

2 Design Overview

We now formalize the READY deployment-qualification problem, which asks whether an agent can be deployed reliably and economically on a concrete enterprise workflow. For a given workflow and agent, READY evaluates the human–AI system induced by an *oversight policy*, π. The policy specifies when the

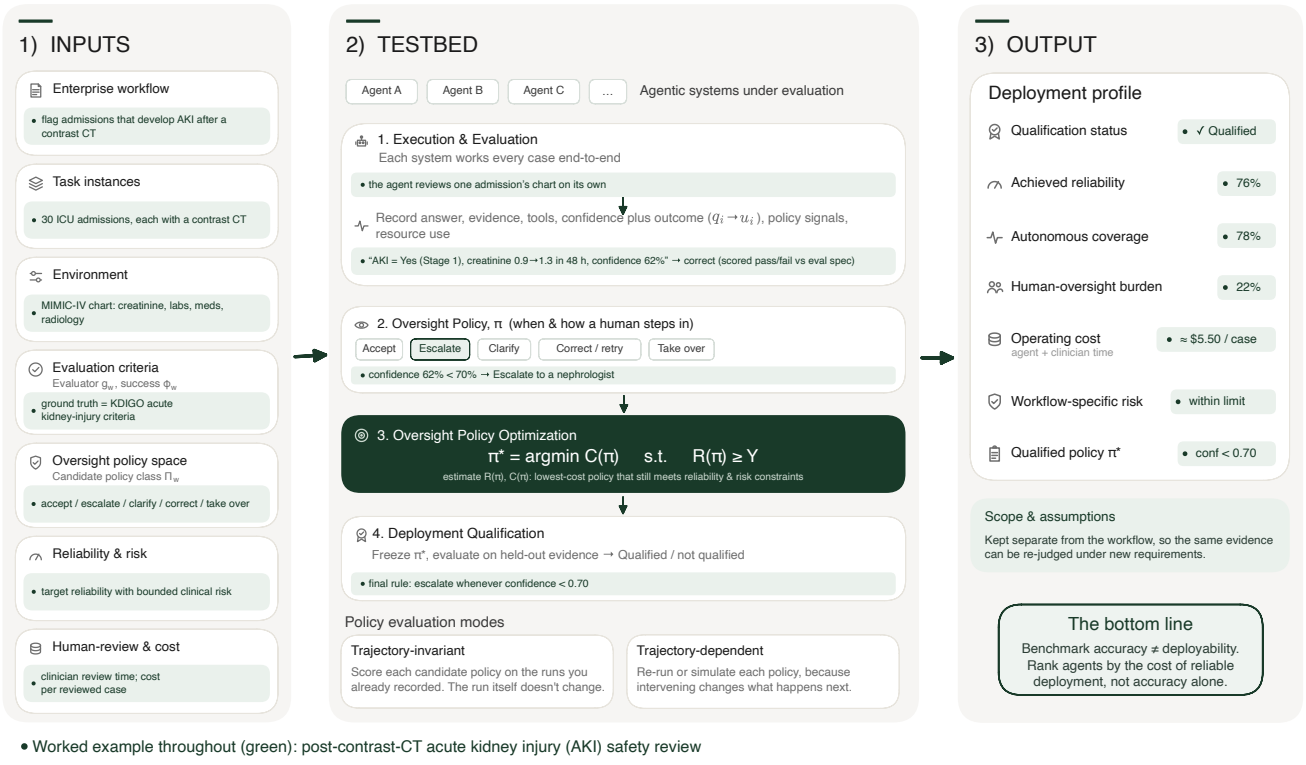

Figure 1. *Reliable Enterprise Agent Deployment* (READY) takes as input a specification defining the workflow, representative task instances, execution environment, evaluation semantics, oversight policy class, and deployment requirements. Within the testbed, multiple agentic systems can be evaluated against the same specification. For each system, READY proceeds through three phases: (1) agent execution and workflow evaluation generate instance-level execution evidence; (2) oversight policy optimization selects the lowest-cost policy satisfying the specified reliability and risk constraints; and (3) statistical qualification freezes the selected policy and evaluates it on held-out cases. The output is a deployment profile for each evaluated system, characterizing the reliability, human-oversight burden, operating cost, and workflow-specific risk supported by the evidence under the stated deployment assumptions.

agent may operate autonomously and when human intervention may be introduced. Depending on the workflow, intervention may occur after a completed result or during execution through review, approval, clarification, correction, retry, escalation, or takeover. At a high level, READY separates three phases:

$$\textbf{Workflow Evaluation} \longrightarrow \textbf{Policy Optimization} \longrightarrow \textbf{Deployment Qualification}.$$

Workflow evaluation defines and measures what constitutes successful execution for a particular form of professional work. Policy optimization uses this evidence to select, from a specified class of oversight policies, the lowest-cost policy that satisfies the deployment requirements. Deployment qualification then freezes the selected policy and statistically evaluates it on held-out evidence. The output of this process is a *deployment profile* describing the reliability–oversight–cost operating point supported by the evidence under the stated deployment assumptions. Figure 1 summarizes the end-to-end process.

2.1 Evaluation Setting and Reliable Deployment Objective

A workflow, w, denotes a reusable type of enterprise work. Let E_w denote the execution environment associated with workflow w, including the tools, data interfaces, systems, and interaction capabilities available during execution. A task instance, x, specifies one concrete case on which the workflow is performed. It contains the inputs made available to the agent, together with the reference material and ground truth required to evaluate that instance. Let $\mathcal{D}_w$ denote the population distribution over task instances of the workflow. An evaluation set, S_w, is a finite sample of instances,

$$S_w = \{x_i\}_{i=1}^{N_w}, \qquad x_i \overset{\text{iid}}{\sim} \mathcal{D}_w. \tag{1}$$

Let m denote the evaluated agent. Executing m on task instance, x_i, in environment, E_w, under oversight policy, π, produces a trajectory:

$$Z_i^\pi = \text{Execute}(m, x_i, E_w; \pi). \tag{2}$$

The trajectory contains the final work product and the observable evidence needed to evaluate the work, including, when relevant, retrieved evidence, tool interactions, intermediate artifacts, interventions, state changes, and resource use. If the agent, environment, or human oversight is stochastic, Z_i^π and all derived quantities are random variables under the corresponding deployment process.

Each workflow defines a workflow-specific evaluator, g_w, that maps a trajectory to a vector of measurements:

$$\mathbf{q}_i^\pi = g_w(Z_i^\pi)\,, \tag{3}$$

where $\mathbf{q}_i^\pi$ may characterize both the final outcome and relevant aspects of the execution process, such as evidence use, adherence to required procedures, or compliance with applicable policies. The workflow also defines a pre-specified *workflow-success predicate*, ϕ_w:

$$u_i^\pi = \phi_w(\mathbf{q}_i^\pi) \in \{0, 1\}, \tag{4}$$

where $u_i^\pi = 1$ indicates that final outcome and any required process or policy conditions are satisfied on instance i. The vector $\mathbf{q}_i^\pi$ retains multidimensional diagnostic information about the execution, while u_i^π is an instance-level execution outcome used to define deployment reliability.

This separation distinguishes measurement from qualification: g_w determines what aspects of the work are measured, while ϕ_w specifies which combination of those measurements constitutes a successful execution. For professional workflows, the success predicate may require a conjunction of outcome correctness with selected process, grounding, or policy requirements, not final-answer correctness alone:

$$Z_i^\pi \xrightarrow{g_w} \underbrace{\mathbf{q}_i^\pi}_{\text{workflow measurements}} \xrightarrow{\phi_w} \underbrace{u_i^\pi}_{\text{instance-level success}} \tag{5}$$

For agent m on workflow w under oversight policy π, we define deployment reliability as

$$R_{m,w}(\pi) = \mathbb{E}_{x\sim\mathcal{D}_w}{}^{1}\left[u^{\pi}(x)\right]. \tag{6}$$

Let $K_w(Z^\pi)$ denote the workflow-specific operating cost associated with an execution under policy π, including agent execution cost and the cost of human oversight. The expected operating cost is

$$C_{m,w}(\pi) = \mathbb{E}_{x\sim\mathcal{D}_w}\left[K_w(Z^{\pi})\right]. \tag{7}$$

READY then casts oversight-policy selection as a constrained optimization problem over a candidate policy class Π_w. Given a target reliability Y, and optionally a workflow-specific risk tolerance B_w, define the feasible set

$$\mathcal{F}_{m,w}(Y, B_w) = \{\pi \in \Pi_w : R_{m,w}(\pi) \geq Y,\ \rho_w(L_w(Z^{\pi})) \leq B_w\}, \tag{8}$$

where L_w is a workflow-specific deployment loss and ρ_w is a corresponding risk functional. The risk constraint is optional; when mean reliability is sufficient, it is inactive, equivalently $B_w = +\infty$. For workflows in which the severity or concentration of failures matters, ρ_w may instead represent a chance constraint or a tail-risk measure such as Conditional Value at Risk (CVaR).

If $\mathcal{F}_{m,w}(Y, B_w) = \varnothing$, no policy in the evaluated policy class supports the requested deployment operating point. Otherwise, READY selects the lowest-cost feasible policy:

$$\boxed{\pi^{\star}_{m,w}(Y, B_w) \in \arg\min_{\pi\in\mathcal{F}_{m,w}(Y,B_w)} C_{m,w}(\pi).} \tag{9}$$

Importantly, Equation 9 defines *policy selection*, not statistical qualification. The policy is selected using development evidence, then frozen and evaluated on held-out qualification data to determine whether the requested reliability and any additional deployment constraints are statistically supported. Section 3 develops the estimation of reliability, operating cost, risk, and statistical qualification in detail.

2.2 End-to-End READY Flow

Figure 1 organizes READY around its inputs, three evaluation phases, and resulting deployment profiles. The input to READY is a specification defining the workflow, its task instances, associated execution environment, oversight policy class and evaluation criteria, including both the workflow-specific evaluator and the instance-level success criterion. Within the READY testbed, a set of agentic systems is evaluated against this specification. For each system, READY executes and evaluates the workflow, optimizes an oversight policy, and statistically qualifies the selected policy on held-out cases. The output is a deployment profile for each evaluated system.

Inputs: specification. READY begins from declarative specifications that define the deployment question to be evaluated. The *workflow specification* identifies the workflow w, task population $\mathcal{D}_w$, sampled evaluation instances S_w, and execution environment E_w. The *evaluation specification* defines the workflow evaluator g_w, the workflow-success predicate ϕ_w, and the observable signals or state available to oversight policies. The *deployment specification* defines the candidate policy class Π_w, target reliability Y,

[1] We simplify the notation here. Strictly speaking, the expectation is taken over both the workflow population $x \sim \mathcal{D}_w$ and any stochasticity in agent execution or human oversight.

human-oversight model, operating-cost model, and any workflow-specific risk tolerance. Keeping these specifications separate allows the same workflow evidence to be interpreted under different deployment requirements.

Phase 1: agent execution and workflow evaluation. READY can evaluate multiple agent systems m against the same deployment specification, producing comparable deployment profiles for each. For each evaluated system m and task instance x_i, READY executes the agent in the workflow environment E_w and records the trajectory Z_i, including the final work product, tool interactions, retrieved evidence, intermediate artifacts, interventions, and resource use. The workflow evaluator maps this trajectory into workflow-specific measurements, $Z_i \xrightarrow{g_w} \mathbf{q}_i$, and the success predicate maps those measurements into an instance-level success outcome, $\mathbf{q}_i \xrightarrow{\phi_w} u_i$. The resulting evidence also includes policy-observable signals and cost measurements needed to evaluate candidate oversight policies.

Phase 2: oversight policy optimization. Using development evidence, READY estimates the reliability, operating cost, and any workflow-specific risk induced by candidate policies in Π_w. It then selects the lowest-cost policy satisfying the specified deployment constraints:

$$\pi^\star \in \arg\min_{\pi \in \Pi_w} C(\pi) \quad \text{subject to} \quad R(\pi) \geq Y,$$

together with any additional risk constraint. When oversight is trajectory-invariant, such as terminal accept-or-escalate review, candidate policies can be evaluated by replaying saved execution evidence. When oversight is trajectory-dependent, intervention changes what happens next, candidate policies must be executed or simulated in the runtime.

Phase 3: deployment qualification. Policy optimization alone does not establish deployability. After the policy $\pi^\star$ is selected, READY freezes it and evaluates the human–AI system on held-out qualification cases not used during policy selection. Deployment qualification is granted only when the held-out evidence statistically supports the specified reliability target and any additional deployment constraints.

Output: deployment profiles. The output of READY is a set of *deployment profiles*, one for each evaluated agentic system. A deployment profile reports the qualified policy, qualification status, achieved reliability and statistical confidence, human-oversight burden, operating cost, workflow-specific risk, and the scope and assumptions under which the profile holds. The deployment profile connects benchmark evidence to an operational decision: whether, and under what oversight policy, the agent can be deployed at the required level of reliability and cost.

Section 3 develops the qualification methodology in detail, while Section 4 describes the evaluation runtime and open interfaces used to implement this architecture across enterprise workflows.

2.3 Running Example: Retrospective Clinical Audit

Throughout the paper, we use a retrospective clinical-audit workflow from CliniCARE-Bench [4] as a running example to instantiate the READY framework.

In CliniCARE-Bench, for each patient-specific case, an agent receives a concise clinical-audit question and access to a longitudinal patient record. It must determine what evidence is needed, retrieve and

reconcile that evidence, apply the relevant clinical or operational standard, and produce an auditable adjudication.

In READY notation, retrospective clinical audit defines the workflow w. A task instance x_i consists of a clinical-audit question paired with the relevant patient record and, where applicable, a particular encounter or clinical event. The environment E_w provides patient-scoped access to structured and free-text EHR data through clinically meaningful retrieval tools, together with facilities for explicit computation and access to governing policy documents.

Executing agent m on case x_i produces a complete evaluation run. The final report contains one of four verdicts—*Yes*, *No*, *Indeterminate: Lack of Data*, or *Indeterminate: Medically Ambiguous*—together with supporting patient- and policy-evidence citations and a stated confidence s_i. The recorded trajectory Z_i additionally contains observable investigation evidence such as tool calls, retrieved information, computations, and intermediate artifacts.

For example, one scenario asks whether an ICU patient developed acute kidney injury within 48 hours after a CT scan. The agent must establish baseline creatinine using a specified hierarchy, identify the CT execution time, inspect the relevant post-CT creatinine trajectory, check chronic dialysis and ESRD (end-stage renal disease) status, apply the KDIGO [10] creatinine criteria, and cite the record evidence supporting the resulting verdict. The evaluation needs to inspect whether the agent retrieved the necessary evidence, used the appropriate temporal window and clinical rule, and avoided unsupported shortcuts. The workflow illustrates why professional-work evaluation may need to measure more than final-answer correctness alone.

The workflow evaluator, g_w, measures multiple aspects of the observable execution, including verdict correctness, appropriate abstention, evidence grounding, policy grounding, and adherence to scenario-specific investigation requirements. The workflow-success predicate, ϕ_w, specifies which of these measurements are required for an execution to count as successful. For example, an illustrative process-aware success definition may take the form

$$u_i = \mathbf{1}\{\text{verdict correct } \wedge \text{ no disqualifying process defect}\}. \tag{10}$$

The clinical-audit case study uses a terminal accept-or-escalate policy class. Here, the agent's stated confidence s_i is recorded as the policy-observable signal, and the oversight policy is defined based on how that signal is used to route a completed adjudication between autonomous acceptance and human review. Confidence is therefore a property of this particular policy instantiation, not a requirement of the general READY framework.

Because the routing decision occurs only after the agent has completed the audit, changing the policy does not change the underlying agent trajectory Z_i. The execution evidence is trajectory-invariant, allowing multiple candidate oversight policies to be evaluated from the same saved runs.

Given a reliability target and deployment assumptions about human-review effectiveness and operating cost, READY uses this evidence to select an oversight policy and then statistically qualify the frozen operating point. The clinical-audit workflow provides a concrete instance of the trajectory-invariant policy-evaluation setting developed more fully in Section 3.

3 Oversight Policy Optimization and Deployment Qualification

Section 2 formulated reliable deployment as a constrained optimization problem over oversight policies. We now develop the quantities, policy-optimization procedures, and statistical qualification methods needed to operationalize that formulation.

The key distinction is between *agent performance* and *deployed-system reliability*. An oversight policy determines how autonomous agent execution is combined with human intervention. Its value depends not only on how often the agent succeeds on its own, but also on how effectively the policy uses observable workflow information to introduce oversight, how effective that oversight is, and what it costs.

We first develop the trajectory-invariant terminal accept-or-escalate setting, where many candidate policies can be evaluated from the same saved execution evidence. We then describe statistical qualification of a selected policy on held-out cases and extend the formulation to trajectory-dependent oversight, where intervention changes the execution trajectory itself.

3.1 Qualification Quantities

For agent m, workflow w, and oversight policy π, let $u^\pi \in \{0,1\}$ denote the workflow-specific success event defined in Section 2.1. The event u^π records whether an individual workflow execution under policy π satisfies the pre-specified success criterion.

We define the reliability of the human–AI system as

$$R_{m,w}(\pi) = \mathbb{E}_{x\sim\mathcal{D}_w}\left[u^\pi(x)\right], \tag{11}$$

where $u^\pi(x) \in \{0,1\}$ denotes whether execution on workflow instance x satisfies the workflow-specific success criterion. The expectation is over workflow instances x drawn from the population represented by $\mathcal{D}_w$ and, when execution is stochastic, over the induced randomness in agent execution, the environment, oversight process, and interacting actors. Since $u^\pi(x)$ is binary, $R_{m,w}(\pi)$ is the probability that an execution under policy π satisfies the workflow-specific success criterion on a randomly drawn workflow instance. Deployment qualification is an aggregate statistical claim about the human–AI system induced by π, not a property assigned to any individual task instance.

Let $K_w(Z^\pi)$ denote the workflow-specific operating cost associated with an execution under policy π. This may include agent execution as well as human review and intervention. The expected operating cost is

$$C_{m,w}(\pi) = \mathbb{E}[K_w(Z^\pi)]. \tag{12}$$

Conceptually, this cost can be decomposed into agent and human components, where both terms may depend on π, as intervention may increase human effort while changing the subsequent agent execution.

$$C_{m,w}(\pi) = C_{\text{agent}}(m,w,\pi) + C_{\text{human}}(m,w,\pi), \tag{13}$$

Equations 11 and 12 define the two canonical quantities in the deployment objective of Equation 9. They are properties of the deployed human–AI configuration under a specified policy, not intrinsic properties of the agent alone.

3.2 Terminal Accept-or-Escalate Oversight

We first consider the trajectory-invariant terminal accept-or-escalate setting used in our clinical-audit case study. In this setting, the agent completes its work before an oversight decision is made. The completed execution provides a result together with an observable routing signal s_i; the policy then determines whether the result is accepted autonomously or routed to a declared human-review path.

To evaluate such policies, READY first executes the agent on representative workflow instances under autonomous execution. For each task instance x_i, this produces a saved trajectory Z_i, an observable routing signal s_i, and an autonomous-success indicator $u_i \in \{0, 1\}$.

Here, (Z_i, s_i, u_i) are evaluation data used for policy qualification; at deployment time, the routing decision is made from observable information such as s_i, while u_i is generally unknown.

Because the terminal routing decision occurs only after Z_i has been generated, the trajectory is invariant to the candidate routing policy. READY can evaluate many accept-or-escalate policies by replaying the same saved evaluation runs instead of re-executing the agent for every candidate policy.

For a scalar routing signal s_i, consider a threshold policy π_τ that accepts the completed result autonomously when $s_i \geq \tau$ and routes it to human review otherwise. We assume that larger values of s_i indicate greater likelihood of successful autonomous execution, so that s_i provides a meaningful ordering of cases for selective routing. The formal routing assumption is given in Appendix A.1. The autonomous coverage under threshold τ is

$$c(\tau) = \Pr(s \geq \tau), \tag{14}$$

and its corresponding human-review rate is

$$e(\tau) = 1 - c(\tau). \tag{15}$$

Coverage is useful in this terminal setting because it is simply the complement of review burden; it is not required as a universal READY deployment metric. Among autonomously accepted cases, define

$$a(\tau) = \Pr(u = 1 \mid s \geq \tau), \tag{16}$$

the success probability of work accepted without review.

Let $a_h(\tau)$ denote the probability that a case routed by policy π_τ is successfully resolved by the declared human-review path. The reliability of the deployed human–AI system is then

$$R(\tau) = c(\tau)a(\tau) + \big(1 - c(\tau)\big)a_h(\tau). \tag{17}$$

The first term corresponds to work accepted autonomously; the second corresponds to work routed to human review.

The review-path performance $a_h(\tau)$ can be either measured from the review process or introduced as a deployment assumption. It is important to note that because routing deliberately selects a non-random subset of cases, $a_h(\tau)$ may vary with the policy: cases routed under a more selective policy may be systematically more difficult than cases routed under another policy. When a constant review-success assumption a_h is available, Equation 17 reduces to

$$R(\tau) = c(\tau)a(\tau) + \big(1 - c(\tau)\big)a_h. \tag{18}$$

This formulation makes clear why autonomous benchmark performance does not determine deployment reliability. Two agents with similar autonomous success rates can support substantially different human–AI operating points because the oversight policies available to them may differ in how effectively they identify cases requiring intervention.

3.3 Cost at a Target Reliability

In terminal accept-or-escalate deployment, the agent has already executed before the routing decision is made. Let k_m denote its expected execution cost per case and let k_h denote the incremental cost of the declared human-review path. Under a constant-cost approximation,

$$C(\tau) = k_m + k_h\, e(\tau) = k_m + k_h\big(1 - c(\tau)\big). \tag{19}$$

For target reliability Y, the population-level policy optimization problem is

$$\tau^\star(Y) = \arg\min_{\tau} C(\tau) \qquad \text{subject to} \qquad R(\tau) \geq Y, \tag{20}$$

together with any additional workflow-specific risk constraint.

Equation 20 defines the ideal population operating point. In practice, $R(\tau)$ and $C(\tau)$ are unknown and policy selection is performed using development evidence. Let $\widehat{R}_{\text{dev}}(\tau)$ and $\widehat{C}_{\text{dev}}(\tau)$ denote the corresponding development-set estimates. A generic development-stage selection rule takes the form

$$\widehat{\tau} \in \arg\min_{\tau} \widehat{C}_{\text{dev}}(\tau) \qquad \text{subject to} \qquad \widehat{R}_{\text{dev}}(\tau) \geq Y_{\text{sel}}, \tag{21}$$

where Y_{sel} is a pre-specified development-stage selection criterion. It may equal the deployment target Y or incorporate a pre-specified margin or other conservative selection rule.

READY does not require a particular optimization algorithm or development-stage selection rule. What is required for qualification is that all policy choices be made without using the held-out qualification cases. The selected policy $\widehat{\pi}$ is subsequently frozen and evaluated against the actual deployment target Y as described in Section 3.5.

When k_m and $k_h > 0$ are constant, minimizing operating cost is equivalent to minimizing human-review burden, or equivalently maximizing autonomous coverage, among feasible policies. Writing

$$e^\star(Y) = e(\tau^\star(Y))\,, \tag{22}$$

the population cost of reliable deployment is

$$C^\star(Y) = k_m + k_h e^\star(Y). \tag{23}$$

This provides an economically meaningful basis for comparing systems: how much operating cost is required for each system to satisfy the same workflow-specific reliability requirement.

The constant-cost model is useful for exposition but is not required by READY. Review effort may vary substantially across cases, and different policies may trigger different forms of review, correction, or escalation. When case-level costs are available, READY estimates $C(\pi)$ directly from the costs induced by the candidate policy instead of assigning a single constant k_h to every reviewed case.

3.4 Quality of the Routing Signal

For terminal accept-or-escalate policies, the value of oversight depends in part on whether the routing signal identifies cases for which autonomous execution is likely to fail. A useful signal should rank higher-risk cases below lower-risk cases, allowing a more selective policy to remove failures from autonomous handling preferentially.

A natural diagnostic is the selective-risk curve,

$$r(\tau) = \Pr(u = 0 \mid s \geq \tau), \tag{24}$$

or equivalently $r(c)$ when operating points are indexed by autonomous coverage. Sweeping the routing threshold traces the risk–coverage relationship: a stronger routing signal achieves lower selective risk at the same autonomous coverage.

We summarize this curve using the area under the risk–coverage curve (**AURC**),

$$\mathrm{AURC} = \int_0^1 r(c)\, dc, \tag{25}$$

with lower values indicating a more favorable risk–coverage tradeoff.

AURC is influenced both by routing quality and by the base autonomous success rate $a_0 = \Pr(u = 1)$. We also report the **AUROC** of the routing signal s for predicting execution success u. AUROC measures ranking discrimination and, unlike AURC, is not directly determined by the prevalence of successful executions.

These metrics are diagnostics for the terminal routing policy class rather than universal READY qualification quantities. Their practical importance is ultimately determined by the operating policies they support: at a fixed reliability target, better routing can reduce the amount of human intervention, and the operating cost, required for qualification.

3.5 Statistical Qualification

Optimizing an oversight policy on development evidence does not establish that the selected policy will satisfy its deployment requirements on future cases. READY separates *policy selection* from *statistical policy qualification*.

The evaluation sample S_w is partitioned by disjoint index sets $I_{\mathrm{dev}}, I_{\mathrm{qual}} \subseteq [N_w]$ into a development set $S_{\mathrm{dev}} = \{x_i : i \in I_{\mathrm{dev}}\}$ and a held-out qualification set $S_{\mathrm{qual}} = \{x_i : i \in I_{\mathrm{qual}}\}$. All choices that determine the policy—including routing rule, threshold, optimization procedure, and any development-stage margin—are made using S_{dev}. The selected policy $\widehat{\pi}$ is then frozen before the qualification outcomes are examined.

For terminal accept-or-escalate oversight, let $\widehat{\tau}$ denote the threshold selected on the development set. For each qualification case i, define

$$d_i = \mathbf{1}\{s_i \geq \widehat{\tau}\}, \tag{26}$$

where $d_i = 1$ denotes autonomous acceptance and $d_i = 0$ denotes human review. The qualification set is used only to estimate the operating characteristics of this fixed policy, not to search for a better threshold.

Measured human-review outcomes. When the outcome of the declared human-review path is observed for each escalated case, let $h_i \in \{0, 1\}$ denote whether that review path produces a successful result. The realized success of the deployed system on qualification case i is

$$v_i = d_i u_i + (1 - d_i) h_i, \tag{27}$$

and the empirical deployment reliability is

$$\widehat{R}(\widehat{\pi}) = \frac{1}{N_{\text{qual}}} \sum_{i \in I_{\text{qual}}} v_i, \qquad N_{\text{qual}} = |I_{\text{qual}}|. \tag{28}$$

In the terminal setting, v_i is the realized policy-level success outcome corresponding to the general success event u_i^{π} in Section 3.1; u_i here denotes the success of the autonomous execution before routing.

When every policy-level outcome is observed and qualification cases are independent, v_i is binary and a one-sided binomial lower confidence bound can be computed directly. At confidence level $1 - \alpha$, READY qualifies the frozen policy only if

$$R_{\text{LCB}}^{1-\alpha}(\widehat{\pi}) \geq Y. \tag{29}$$

A point estimate above the target is not sufficient: the held-out evidence must support a reliability lower bound that itself clears the deployment requirement.

Human-review performance as a deployment assumption. In retrospective benchmark analysis, the declared human-review path may not be executed for every escalated case. Its effectiveness can instead be introduced explicitly as a deployment assumption. Suppose, for example, that every routed case is assumed to be successfully resolved with probability a_h. For each qualification case define the policy-level conditional success value

$$y_i(a_h) = d_i u_i + (1 - d_i) a_h \in [0, 1]. \tag{30}$$

Under the constant-review assumption, the population reliability of the fixed policy is

$$R(\widehat{\pi}; a_h) = \mathbb{E}\left[y_i(a_h)\right], \tag{31}$$

This quantity is no longer a simple binomial proportion, because reviewed cases contribute an assumed success probability rather than an observed binary outcome. Writing n_{acc} for the number of accepted qualification cases and $\widehat{p}_{\text{acc}}$ for the observed success rate among them, the empirical estimate can be written as

$$\widehat{R}(\widehat{\pi}; a_h) = \frac{n_{\text{acc}}}{N_{\text{qual}}} \widehat{p}_{\text{acc}} + \frac{N_{\text{qual}} - n_{\text{acc}}}{N_{\text{qual}}} a_h, \tag{32}$$

in which only $\widehat{p}_{\text{acc}}$ is estimated from observed outcomes.

For statistical qualification, however, both accepted-case success and the fraction of cases routed by the frozen policy vary across samples from the workflow population. READY constructs a one-sided lower confidence bound directly for the mean of the bounded case-level quantities $\{y_i(a_h)\}_{i \in I_{\text{qual}}}$ rather than treating the observed routing fraction as fixed. Denote this bound by

$$R_{\text{LCB}}^{1-\alpha}(\widehat{\pi}; a_h). \tag{33}$$

The policy qualifies conditionally on the declared human-review assumption only if

$$R_{\text{LCB}}^{1-\alpha}(\widehat{\pi}; a_h) \geq Y. \tag{34}$$

The lower bound may be computed using a pre-specified valid procedure for the mean of bounded case-level outcomes. When cases are clustered, the qualification procedure operates at the corresponding cluster level as described below.

No statistical confidence is attributed to the assumed term. The qualification statement is explicitly conditional, of the form "qualified at target Y under the declared review assumption a_h." For policy families whose contributions do not admit this decomposition, the same conditional bound is obtained by resampling qualification cases and re-evaluating the fixed policy on each resample. If a_h is itself estimated from separate human-review data, uncertainty in that estimate must also be propagated rather than treating a_h as known exactly; a conservative qualification replaces a_h with a one-sided lower confidence bound $a_{h,\text{LCB}}$ in Equation 34.

Break-even review assumption. Because qualification reliability $R_{\text{LCB}}^{1-\alpha}(\widehat{\pi}; a_h)$ is nondecreasing in a_h, the dependence on the human-review assumption can be summarized by a single quantity: the weakest review effectiveness under which the frozen policy still qualifies:

$$a_h^{\min}(\widehat{\pi}, Y) = \inf \left\{ a_h \in [0,1] : R_{\text{LCB}}^{1-\alpha}(\widehat{\pi}; a_h) \geq Y \right\}. \tag{35}$$

The deployment profile can then state an auditable requirement on the human process: the selected policy qualifies at target Y provided the declared review path achieves effectiveness at least $a_h^{\min}$.

If no $a_h \in [0,1]$ satisfies the criterion, the policy is reported as not qualified at target Y under any constant review-success assumption.

Sampling unit. The statistical sampling unit must match the unit over which the deployment claim is intended to generalize. Multiple stochastic executions of the same task instance are not treated as independent new workflow cases. When repeated runs are present, they are carried together at the task-instance level.

Similarly, when cases are naturally clustered by patient, customer, account, site, or another higher-level unit, qualification must preserve that dependence structure, for example by constructing confidence intervals or resampling at the cluster level.

A policy that fails the statistical criterion is reported as *not qualified at target* Y or as having *insufficient evidence*; it is not retuned using the qualification set. Any change to the policy $\widehat{\pi}$ after examining qualification outcomes requires a new held-out qualification evaluation.

Unless otherwise stated, READY qualification claims are *pointwise* for a pre-specified deployment target and frozen policy. Simultaneous statistical claims across multiple targets, policies, or operating points require an appropriate simultaneous-inference procedure.

3.6 Risk-Sensitive Qualification

Reliability treats every unsuccessful execution as the same binary event. For workflows in which failures differ materially in consequence, READY may additionally define a workflow-specific deployment loss

$$L_w(Z^\pi) \geq 0 \tag{36}$$

and impose the constraint

$$\rho_w(L_w(Z^\pi)) \leq B_w, \tag{37}$$

where B_w is the declared risk tolerance and ρ_w is a workflow-specific risk functional. Examples include a bound on the probability of a specified severe event or a tail-risk measure such as Conditional Value at Risk (CVaR).

This constraint is optional. The common READY objective remains reliable deployment at minimum operating cost; an additional risk constraint is introduced when binary success alone does not adequately represent the consequences of failure. Estimation and statistical qualification of risk-sensitive constraints are discussed further in Appendix A.2.

3.7 Trajectory-Dependent Oversight

The terminal formulation above is especially convenient because the routing decision does not change the saved agent trajectory. More general enterprise workflows allow intervention during execution.

Let h_t denote the observable workflow history at decision point t, and let

$$\pi(o_t \mid h_t) \tag{38}$$

denote an oversight policy over intervention decisions o_t, such as approval, correction or takeover. Executing the workflow under π induces a policy-dependent trajectory

$$Z_i^\pi \sim P(\,\cdot \mid m, x_i, E_w, \pi\,). \tag{39}$$

The qualification objective remains the one defined in Equation 9. The difference is how the evidence needed to estimate that objective is generated.

For trajectory-invariant terminal policies, candidate policies can be evaluated by replaying saved trajectories under different routing rules. For a trajectory-dependent policy, changing π changes what the agent, oversight actor, or interacting user subsequently observes and does. Reliability, cost, and risk must instead be estimated by executing or appropriately simulating the candidate policy in the evaluation runtime.

The phases introduced in Section 2.2 should therefore be understood as a logical evaluation flow rather than necessarily a one-pass computation. For trajectory-dependent oversight, policy optimization may repeatedly invoke agent execution and workflow evaluation while searching over candidate policies.

Policy-in-the-loop evaluation may change both sides of the deployment tradeoff. Intervention can improve the probability of successful completion, but can also change agent execution length, human effort, recovery behavior, and the distribution of subsequent outcomes. These effects are captured by the same general quantities $R_{m,w}(\pi)$, $C_{m,w}(\pi)$, and, when applicable, $\rho_w(L_w(Z^\pi))$.

This setting is naturally connected to simulation-based and black-box optimization, where objective and constraint values need not admit closed-form expressions and can instead be estimated through stochastic simulation or system execution [11, 12]. READY does not prescribe an optimization algorithm. The policy representation and optimization method may be workflow-dependent. What is shared across workflows is the deployment objective and qualification procedure: candidate policies are evaluated in terms of the reliability, cost, and risk of the resulting human–AI system, a policy is selected using development evidence, and the frozen policy is qualified on held-out evidence.

Empirical instantiation. Section 5 applies the qualification methodology above to CliniCARE-Bench. The analysis is conducted on the benchmark's full four-way adjudication space using a signal-based accept-or-escalate policy that is agnostic to the class labels, with sensitivity analyses in Appendix B and a cost-of-conflation control in Appendix B.5.

4 READY Evaluation Runtime and Open Testbed

Sections 2 and 3 define what READY qualifies and how an oversight policy is selected. This section describes the execution and interface testbed that produces the evidence required by that methodology.

READY does not introduce a new general-purpose agent-evaluation harness. Its reference implementation uses Inspect AI [9] for task execution, agent integration, sandboxed environments, scoring, intervention, and evaluation logging. READY adds the workflow-level semantics and deployment-qualification layer needed to translate executions into a reliability–oversight–cost deployment profile.

4.1 Runtime Layering and Reference Execution Substrate

The implementation follows the three-layer architecture in Figure 1. Non-executing *specification artifacts* define the workflow, evaluation semantics, and deployment assumptions. The *evaluation runtime* instantiates these specifications and executes the agent, workflow environment, and any required oversight or user actors. The resulting execution evidence is evaluated and passed to the qualification methodology in Section 3.

READY standardizes the semantics and evidence that cross these boundaries without requiring every evaluation to share a task representation, agent implementation, environment technology, or grading mechanism, so existing evaluation infrastructure can be reused.

In the Inspect reference implementation, an executable evaluation is represented by an Inspect `Task`, which combines a collection of `Samples` with an agent or `Solver`, one or more `Scorers`, and the required runtime configuration. READY introduces the workflow and deployment semantics above these execution objects. The correspondence is an implementation mapping, not a one-to-one semantic equivalence.

4.2 Specification Interfaces

The runtime consumes three logically distinct classes of specification artifacts corresponding to Figure 1.

Workflow specification. The workflow specification identifies the reusable workflow, w, the task-instance population, $\mathcal{D}_w$, from which the evaluation sample, S_w, is drawn, and the environment specification, E_w. It defines the work to be performed and the population for which the evaluation provides evidence. A concrete task instance, x_i, supplies the case-specific inputs for one execution.

READY does not require a second executable task format. In the Inspect reference implementation, a READY task instance is represented by an Inspect `Sample`, and a collection of task instances is represented by an Inspect `Dataset`. The enclosing Inspect `Task` specifies how those samples are executed and evaluated. This distinction is important: an READY *task instance* corresponds to an Inspect `Sample`, not to an Inspect `Task`.

Evaluation specification. The evaluation specification defines the workflow evaluator, g_w, the workflow-success predicate, ϕ_w, and any routing signal exposed to an oversight policy.

It specifies the transformation $Z_i^\pi \xrightarrow{g_w} \mathbf{q}_i^\pi \xrightarrow{\phi_w} u_i^\pi$ of Equation 5.

The mechanisms used to compute $\mathbf{q}_i^\pi$ may be deterministic, human-annotated, model-assisted, or composed from multiple evaluation mechanisms. In Inspect, these computations can be implemented through one or more `Scorers`, including externally computed scores when appropriate. READY's evaluator, g_w, denotes the workflow-level semantics of what is measured; an Inspect `Scorer` is one executable mechanism for producing those measurements. The success predicate, ϕ_w, remains an READY-level definition that determines which measurements constitute acceptable execution for deployment qualification.

Deployment assumptions. Deployment assumptions remain separate from measured evaluation evidence. They include the reliability target, Y, the human-oversight model, operating-cost assumptions, and any workflow-specific risk tolerance. Different organizations may therefore interpret the same measured execution evidence under different deployment requirements.

The development and held-out qualification partitions required by Section 3.5 are versioned with the READY evaluation configuration. Instances used to select an oversight policy must remain separate from the instances used to qualify the frozen policy.

Table 1 summarizes the minimal logical interface required of an READY-compatible runtime and its reference realization in Inspect.

4.3 Execution, Actors, and Policy Evaluation

The evaluation runtime instantiates the workflow environment and executes agent m on task instance x_i under oversight policy π. The runtime is responsible for making the workflow-specified tools, data, systems, and interaction interfaces available while preserving the declared evaluation conditions.

Inspect provides the reference execution mechanisms for this layer. Agents may be implemented through native Inspect agents or solvers, or integrated from external agent frameworks through the Inspect Agent Bridge. Workflow-specific files and computational environments can be attached to individual samples and executed in controlled sandboxes. These implementation choices do not change the READY semantics of the workflow or trajectory.

For workflows requiring interaction, READY distinguishes two additional actor roles. An *oversight actor* represents an organization-side reviewer or operator who may approve, clarify, correct, reject, escalate,

READY interface	Required semantics	Inspect reference realization
Task population	Represent the sampled workflow instances S_w, $x_i \sim \mathcal{D}_w$, with stable identifiers and required case inputs.	`Dataset` of `Samples`
Agent execution	Execute agent m on x_i under the declared model, harness, and runtime configuration.	`Agent` or `Solver`; external agents through Agent Bridge
Workflow environment	Expose the tools, data, files, systems, and computational state specified by E_w.	Task/sample tools and sandbox configuration
Oversight interaction	Allow policy π to approve, reject, clarify, intervene, escalate, or invoke another actor when the workflow requires it.	Approval policies, approver, agent intervention, or task-specific actors
Execution evidence	Preserve the observable evidence required to interpret Z_i^π, including the final work product and relevant interaction history.	Per-sample evaluation log, transcript, outputs, metadata, and retained artifacts
Workflow evaluation	Compute $\mathbf{q}_i^\pi = g_w(Z_i^\pi)$ and retain the measurements used by the deployment criterion.	One or more `Scorers` or externally supplied scores
Qualification inputs	Expose u_i^π, routing signal s_i, resource/cost measurements, and the sampling identifiers required by READY qualification.	Scores, sample metadata, transcript/log fields, usage measurements, and READY-derived records
Evaluation lineage	Identify the workflow, case, agent, model, environment, evaluator, policy, and execution configuration associated with the evidence.	`Task/Sample` configuration and versioned evaluation logs

Table 1. Minimal logical interface for an READY-compatible evaluation runtime. READY defines the semantics required for deployment qualification; the rightmost column shows their reference realization in Inspect AI. Other execution substrates may implement the same interface using different native abstractions.

or take over work. A *user actor* represents an external user or counterparty interacting with the deployed system. Either actor may be human or simulated, but its identity, configuration, and available actions must be part of the evaluation configuration.

Inspect approval policies and agent-intervention mechanisms provide reference implementations for several forms of oversight. For example, approval policies can require selected tool calls to be approved, rejected, or escalated, while agent intervention permits a human operator to interrupt or redirect an executing agent. These mechanisms do not define the READY policy class Π_w; they are runtime primitives through which particular policies may be implemented. Other forms of interaction may be realized through custom agents, solvers, or workflow-specific simulators.

The runtime supports both policy-evaluation procedures introduced in Section 2.2.

For *trajectory-invariant* policies, the relevant agent execution is independent of the terminal routing decision. The runtime can therefore execute the agent once, preserve the resulting evidence, and evaluate multiple candidate policies by replaying the same completed executions. The accept-or-escalate analysis in Section 3.2 uses this path.

For *trajectory-dependent* policies, intervention changes what happens next. The candidate policy must participate in execution, and the runtime records the resulting policy-dependent trajectory Z_i^π. Reliability, cost, and risk for that policy must be estimated from executions or simulations performed under the policy itself.

4.4 Evaluation Evidence and Workflow Evaluation

Each execution produces a sample-level evaluation record containing the observable evidence required by the workflow evaluator and qualification analysis. READY does not introduce a separate universal trajectory format. Instead, Z_i^π denotes the workflow-relevant observable execution trajectory represented by the underlying runtime record.

In the Inspect reference implementation, evaluation logs preserve sample-level outputs, model and tool interactions, scoring information, metadata, usage information, and the execution transcript. For bridged agents, model interactions are likewise captured in the Inspect transcript. When human or automated interventions occur during execution, those interventions should remain part of the retained evidence.

The workflow evaluator applies g_w to this record to produce $\mathbf{q}_i^\pi$, and the pre-specified predicate ϕ_w derives the corresponding success event u_i^π. Diagnostic measurements remain available independently of this binary success event. For example, a workflow may retain outcome correctness, evidence grounding, process adherence, appropriate abstention, and resource use even when only a subset of those measurements enters ϕ_w.

Inspect's scoring interface supports multiple workflow measurements without requiring them to be collapsed into a single benchmark score. Where the saved execution record contains sufficient evidence, an updated evaluator can also be applied to an existing evaluation log without re-executing the agent.

For workflows requiring repeated stochastic execution, the same logical task instance may be executed for multiple epochs. READY retains the underlying task-instance identity and epoch-level evidence so that repeated runs are not treated as independent task instances during statistical qualification.

Running example: CliniCARE-Bench. For CliniCARE-Bench, each case is represented as an evaluation sample whose execution record contains the completed clinical adjudication, supporting patient- and policy-evidence citations, the case-level routing signal, and the observable investigation trajectory, including retrieved evidence, tool calls, computations, and intermediate artifacts. The CliniCARE evaluator derives verdict and process measurements from this record and applies the workflow-specific success predicate before the resulting evidence is passed to READY qualification.

4.5 Qualification Handoff and Deployment Profile

The boundary between workflow evaluation and deployment qualification is an evidence interface rather than a new task evaluator. The qualification layer consumes measured execution evidence together with declared deployment assumptions and applies the methodology in Section 3.

For the trajectory-invariant accept-or-escalate case, the minimal qualification record for task instance i contains the workflow-defined success event u_i, the observable routing signal s_i, and the execution or resource measurements required by the cost model. It also retains the identifiers needed to associate the record with its task instance, statistical sampling unit, evaluated system, and versioned evaluation configuration. These quantities can be extracted from Inspect scores, metadata, usage measurements, and evaluation logs into the READY qualification dataset.

Versioned qualification record. The qualification record has a stable, versioned schema so that a deployment claim can be reproduced and audited. Each record carries a `schema_version`; a `lineage` block that version-stamps the workflow, task population, evaluated system, environment, evaluator g_w, success predicate ϕ_w, and routing-signal definition; the per-instance `measurements` (u_i, s_i, retained diagnostics $\mathbf{q}_i$, and cost); and the identifiers of the task instance and statistical sampling unit. Deployment assumptions are recorded separately with the profile, not in the per-instance record, so the same records support multiple deployment analyses.

The development and held-out partitions carry an identical schema; only the `partition` field differs. Deployment assumptions (Y, a_h, review cost) are attached to the resulting profile, keeping measured evidence separate from the assumptions under which it is interpreted.

Section 3.5 then selects the oversight policy on development data and qualifies the frozen operating point on held-out evidence. The qualification computation is intentionally separate from Inspect's task scoring: Inspect records what happened and how the workflow evaluator scored it, while READY determines which deployment policy is supported by that evidence.

For trajectory-dependent policies, the same logical handoff applies, but the measurements correspond to trajectories generated under the candidate policy π, and the retained evidence includes the policy and actor configuration needed to interpret Z_i^π, u_i^π, and its associated operating cost.

The output is the *deployment profile* defined in Section 2.2: a statistically supported operating point conditional on the workflow, evaluated system, oversight policy, and declared deployment assumptions, rather than a context-free model score.

4.6 Contributing and Maintaining Evaluations

New enterprise workflows can be added to READY without changing the qualification methodology or the reference execution infrastructure. A contribution supplies an executable workflow evaluation together with the specification artifacts of Section 4.2: the workflow identity and represented population, the evaluation semantics g_w and ϕ_w, any routing signal, the development and held-out partitions, and the measurements needed for cost analysis. The same interface serves public benchmarks and enterprise-internal evaluations; READY does not require proprietary data or protected references to be made public, only that the workflow, evaluation, and reported deployment claim remain explicit.

Validation. Interface conformance establishes interoperability; validation checks whether the resulting evidence supports a meaningful deployment claim. A reviewer independent of the contributor checks four aspects against contributor-supplied evidence: *workflow validity* (a recognizable form of enterprise work sampled from the stated population), *evaluation validity* (g_w and ϕ_w reproduce success labels within a pre-registered tolerance from observable evidence), *execution validity* (reproducible dry runs with no hidden-reference access or unintended shortcuts), and *qualification validity* (a frozen development/held-out partition with policy selection never touching held-out data). Verdicts are recorded with the evaluation lineage so that acceptance is reproducible rather than a matter of reviewer opinion.

Versioned evaluation lineage. A READY result is tied to a versioned *evaluation lineage* rather than just a benchmark or model name. The lineage pins the workflow and task population, agent and model configuration, execution environment, evaluator g_w and predicate ϕ_w, routing signal, and the evidence used for qualification, through concrete build identifiers such as commit hashes, dependency and image

digests, and the seeds governing sampling and any split. A qualification claim then has the form

$$\text{evaluation lineage} \;+\; \text{deployment assumptions} \;\longrightarrow\; \text{deployment profile},$$

separating the evidence that was measured from the assumptions under which it is interpreted. Corrections and later releases create new evaluation records rather than silently reinterpreting earlier profiles.

Maintenance and requalification. Because lineage components are versioned separately, different changes have different consequences:

- **Assumptions change** $\rightarrow$ *recompute*. When only deployment assumptions change (target reliability, review model, or cost), the agent evidence is unchanged and READY recomputes the oversight policy and profile from existing records.
- **Evaluation changes** $\rightarrow$ *re-evaluate* when evidence is sufficient. When g_w or ϕ_w changes, preserved trajectories may be re-scored, but only when each trajectory records every input the revised evaluator reads; otherwise the case is re-executed.
- **Trajectory or population changes** $\rightarrow$ *re-execute*. Material changes to the agent, harness, environment, tools, or represented population can alter the trajectory or the claim's scope, so the agent is rerun and passed through the same qualification procedure as a new evaluation.

READY thus treats deployment qualification not as a permanent property of an agent but as an evidence-backed claim attached to a specific lineage and deployment context.

5 Empirical Evaluation: Deployment Qualification on CliniCARE-Bench

We use CliniCARE-Bench to illustrate the deployment-qualification methodology of Section 3. The goal is not to reproduce the full CliniCARE benchmark, but to measure the quantities that matter for READY: whether a routing signal supports selective autonomy, how much human oversight a given reliability target demands, and whether systems with similar autonomous performance differ in deployment value. We organize the section around three questions.

1. Does the routing signal separate more reliable from less reliable autonomous executions? (Section 5.3)
2. How does the required human-review burden grow as the reliability target increases? (Section 5.4)
3. Can systems with similar autonomous performance support materially different qualified operating points? (Sections 5.5–5.6)

5.1 Experimental Setup

Adapting CliniCARE-Bench to READY. Adapting this existing dataset required no change to its clinical content, only expressing it through the interface of Section 4: each adjudication case becomes a task instance x_i (an Inspect `Sample`); the patient-scoped EHR retrieval tools and sandbox become the environment E_w; the four-way verdict grader becomes the evaluator g_w with success predicate ϕ_w; the report's stated confidence becomes the routing signal s_i; and the fixed case split becomes the development and qualification partitions. Deployment assumptions (Y, a_h, cost) stay separate, so the same evidence supports the sensitivity analyses of Appendix B. The four reference verdicts are

$$\mathcal{Y} = \{\text{Yes}, \text{No}, \text{Indeterminate: Lack of Data}, \text{Indeterminate: Medically Ambiguous}\}. \tag{40}$$

As established in Section 2.3, the two indeterminate outcomes are task-level clinical judgments—a correct indeterminate verdict is itself a successful execution—and are distinct from the READY decision to escalate a completed case for review. Keeping them separate lets abstention be handled inside the common READY optimization without treating every indeterminate output as a failure or a forced escalation.

Success indicators. Let $y_i^\star \in \mathcal{Y}$ denote the reference adjudication, $\widehat{y}_i$ the agent adjudication, and s_i the routing signal. The primary four-way success indicator is

$$u_i = \mathbf{1}\{\widehat{y}_i = y_i^\star\}. \tag{41}$$

and a process-aware variant, used as a robustness check in Appendix B, additionally requires the absence of a disqualifying process defect,

$$u_{i,\text{df}} = \mathbf{1}\left\{\widehat{y}_i = y_i^\star \ \wedge \ \text{no disqualifying process defect}\right\}, \tag{42}$$

Systems, cases, and labels. We evaluate 16 systems on all 750 cases, giving 12,000 system–case runs. Reference verdicts follow the natural cohort distribution rather than a balanced one: *Yes* 352 (46.9%), *No* 248 (33.1%), *Indeterminate: Lack of Data* 123 (16.4%), and *Indeterminate: Medically Ambiguous* 27 (3.6%).

Routing signal. The signal s_i is the confidence each system states in its own report (required as a percentage), which we rescale to $[0, 1]$ and observe across the full range. It is present for 11,992 of 12,000 runs; the eight exceptions, seven stating no confidence and one producing no report, are assigned $s_i = 0$ and escalated first. Systems differ sharply in signal resolution, from 5 distinct values (Gemini-3.1-Pro) to 36 (GPT-5.4-mini). This granularity is itself deployment-relevant: a system with few distinct values has correspondingly few reachable operating points.

Development and qualification split. Cases are partitioned once, uniformly at random with a fixed seed, into $|S_{\text{dev}}| = 375$ and $|S_{\text{qual}}| = 375$. The split is over *cases*, not runs, so every system is developed and qualified on an identical cohort and all cross-system comparisons are paired. The draw preserves verdict proportions without stratification (S_{dev}: 172/126/64/13; S_{qual}: 180/122/59/14).

Human-review assumption. Because CliniCARE does not execute the declared review path for every routed case, we treat human-review success a_h as a deployment assumption, with primary value

$$a_h = 0.9, \tag{43}$$

and sensitivity to alternatives in Appendix B. All qualification statements are therefore conditional on this assumption, and we also report the break-even value $a_h^{\min}$ (Equation 35), the weakest review success under which a policy still qualifies. The assumption also imposes a ceiling: because deployment reliability is a convex combination of autonomous and reviewed outcomes and selective success $1 - r(\tau)$

stays below a_h at every coverage level we observe, no policy can exceed reliability $a_h = 0.90$. Targets above 0.90 are unreachable here not because of agent capability but because of the assumed quality of the review path, a structural point a capability-only benchmark cannot express.

5.2 Escalation Policy

We evaluate a single terminal accept-or-escalate policy that acts only on the routing signal, selected on S_{dev} and frozen before qualification. Because the workflow is treated as a four-way classification with success indicator u_i, the policy is deliberately agnostic to what the classes *mean*: class semantics enter only through the success predicate ϕ_w, never through the escalation rule. The same threshold applies to all four predicted adjudications,

$$\pi_\tau(s_i) = \begin{cases} \text{accept}, & s_i \geq \tau, \\ \text{review}, & s_i < \tau. \end{cases} \tag{44}$$

Fitting τ to hit Y directly on a sampled development set is prone to selection bias and tends to fail the stricter held-out lower-bound test. We instead optimize against a margin-adjusted target,

$$\tau^\star(Y) = \arg\min_\tau C(\tau) \qquad \text{subject to} \qquad \widehat{R}_{\text{dev}}(\tau; a_h) \geq Y + \delta, \tag{45}$$

where $\widehat{R}_{\text{dev}}(\tau; a_h)$ is the development-partition reliability estimate under the declared review assumption. All CliniCARE policies use a fixed margin $\delta = 0.05$.

5.3 Autonomous Performance and Routing Quality

We first ask whether s_i carries information about autonomous success beyond the aggregate task score. For each system we sweep the threshold (Equation 44) over the development data and compute the autonomous coverage $c(\tau)$ and selective risk $r(\tau)$ of Section 3.4. Figure 2 shows the resulting risk–coverage curves. An informative signal should lower selective risk as coverage falls, because the first cases removed from autonomous handling should be disproportionately failures, which is broadly what we observe. Each curve begins at the system's lowest reachable coverage; the Gemini models return 100% confidence on up to 73% of cases, so even their most selective reachable policy still leaves most runs autonomous.

Table 2 summarizes autonomous success and routing quality. Across the 16 systems, base accuracy a_0 and AUROC are essentially uncorrelated (Pearson $\rho = -0.12$): a system can be accurate yet rank its own outputs poorly, or the reverse. Qwen-3.7-Plus is the clearest case, pairing one of the lowest base accuracies with the best AUROC. AURC, by contrast, tracks base accuracy closely ($\rho = -0.80$ with a_0), which is exactly why we report both—AURC captures the absolute risk–coverage tradeoff a deployment faces, while AUROC isolates ranking quality from the base rate.

5.4 Reliability–Oversight Frontier

Routing quality becomes operational when translated into the oversight required to meet a reliability target. For each target Y we solve Equation 45 on S_{dev} to select the highest-coverage threshold meeting the requirement, freeze it, and apply it to the held-out S_{qual}. Figure 3 plots the resulting human-review burden against target reliability. This READY deployment frontier shows how much work must be routed to review as the organization demands more reliable operation, and it lets systems be compared

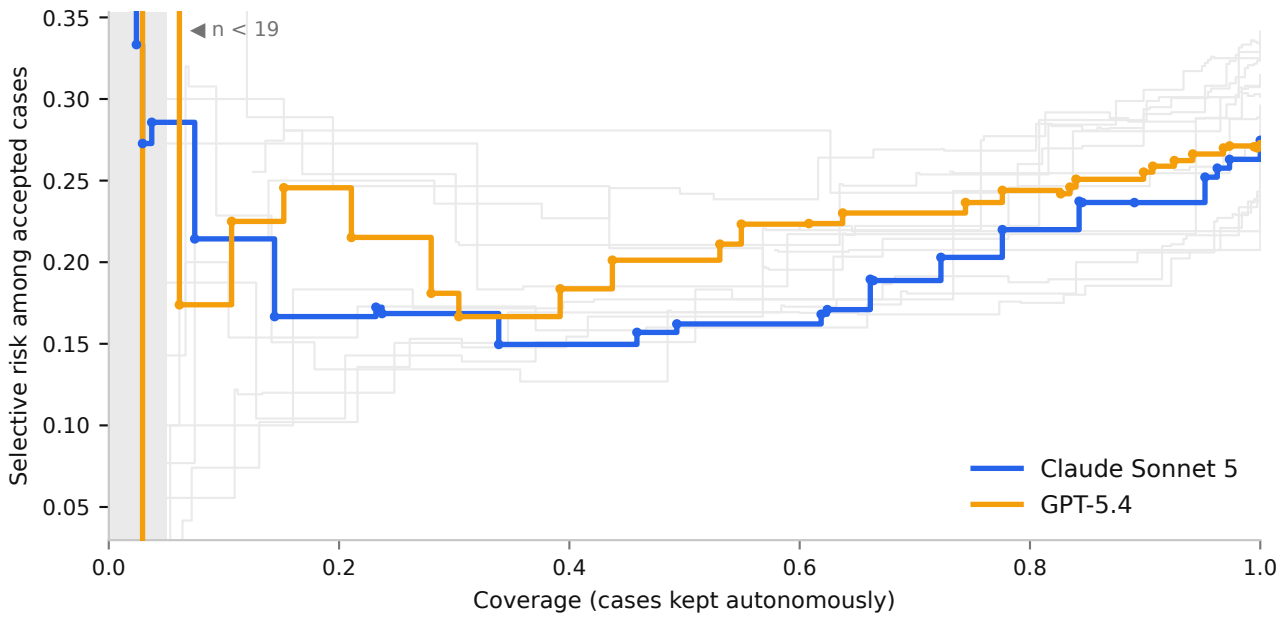

Figure 2. Risk–coverage behavior of the stated-confidence routing signal on the development partition. Sweeping the routing threshold traces the tradeoff between autonomous coverage and failure risk among autonomously accepted cases; lower curves indicate more effective selective routing. Faint curves show all 16 systems; highlighted are the running-example pair, Claude Sonnet 5 and GPT-5.4, whose autonomous accuracies differ by only 0.3 percentage points. Markers on the highlighted curves denote their reachable operating points: only a threshold at a tie-group boundary of a system's stated confidences can be selected. The shaded band marks coverage levels supported by fewer than 19 accepted cases, where the selective-risk estimate is unstable.

at a common deployment requirement rather than a common coverage level: at a fixed target, the system needing less review supports the more efficient policy under the same review assumption.

5.5 Held-Out Qualification and Deployment Comparison

READY separates a development-set operating point from a statistically qualified deployment claim: we evaluate each frozen policy on S_{qual} and compute the reliability estimate $\widehat{R}$ together with its one-sided 95% lower confidence bound (LCB) from Section 3.5. Following Equation 34, the LCB attaches exact Clopper–Pearson confidence to the observed accepted component and *no* confidence to the assumed review term a_h; the policy qualifies iff LCB $\geq Y$. The step-by-step computation is given in Appendix B.1. Table 3 reports qualification at $Y = 0.76$.

The table separates two ways a point-estimate optimum fails to become a deployment. GLM-5.2 requires the lowest review burden in the table (10.9%) and its point estimate clears the target ($\widehat{R} = 0.762$), yet it does not qualify. Because it accepts 89% of cases, almost all of its reliability rests on the measured component, giving it the widest confidence interval in the table ($\widehat{R} - \text{LCB} = 0.037$), while its point estimate clears Y by only 0.002. The same low review burden also leaves the review assumption without leverage: raising a_h moves only the 10.9% of mass that is reviewed, so no $a_h \in [0, 1]$ rescues it. Making GLM-5.2 qualify would require $\delta \geq 0.075$, raising its review burden to 32.0%—above Opus 5 and GPT-5.5. At the other extreme, Gemini-3.1-Pro—whose stated confidence takes only five distinct values—can reach no development-feasible threshold below full review, so it qualifies only by routing 100% of cases to a human, attaining exactly a_h without deploying. Together these cases make the central point: optimizing

System	a_0	AURC ↓	AUROC ↑	Succ@50%
`Claude Code`				
Opus 5	75.7	0.169	0.680	83.1
Sonnet 5	72.5	0.203	0.681	83.7
`Codex`				
GPT-5.6-Sol	73.1	0.189	0.640	81.8
GPT-5.6-Luna	69.9	0.195	0.645	79.9
GPT-5.5	75.7	0.172	0.652	83.7
GPT-5.4	72.8	0.222	0.601	79.2
GPT-5.4-mini	67.5	0.248	0.628	75.9
`Gemini CLI`				
Gemini-3.6-Flash	71.7	0.223	0.619	79.3
Gemini-3.5-Flash	70.4	0.235	0.618	78.0
Gemini-3.1-Pro	70.9	0.261	0.563	74.5
`opencode`				
DeepSeek-V4-Pro	68.5	0.229	0.664	79.2
DeepSeek-V4-Flash	68.8	0.263	0.602	75.4
GLM-5.2	75.7	0.168	0.652	82.8
Qwen-3.7-Plus	66.4	0.219	0.711	79.8
MiniMax-M3	66.4	0.255	0.676	79.8
Kimi-K2.7-Code	65.9	0.261	0.661	76.1

Table 2. Autonomous performance and routing quality on the development partition. $n = 375$ cases per system. a_0 is four-way verdict accuracy under full autonomy, with no review. AURC summarises the risk–coverage curve of Figure 2 and is the quantity the deployment frontier is built from; lower is better. AUROC measures ranking quality independent of baseline accuracy; higher is better. Succ@50% is selective success over the half of cases each system is most confident in.

a policy on fixed data is easy; making a deployment claim that survives on new data is a much stronger bar.

Because Review% at a single target is a step function of each system's reachable threshold grid, systems with coarse signals overshoot—$\widehat{R} - Y$ ranges from $+0.002$ (GLM-5.2) to $+0.140$ (Gemini-3.1-Pro). The margin is what closes the gap between a point estimate and a qualified deployment: without it only 46% of policies qualify across the frontier sweep, against 95% with it. Cross-system review burden is therefore best compared on the full frontier of Figure 3, not at a single target.

5.6 Similar Autonomous Performance, Different Deployment Value

The central READY hypothesis is that autonomous benchmark performance alone does not determine deployment value. To test it, we compare two systems with nearly identical full-autonomy accuracy but different routing quality, with the pair chosen on S_{dev} and frozen before we read the held-out results.

A capability leaderboard would rank GPT-5.4 ($a_0 = 72.8$) and Sonnet 5 ($a_0 = 72.5$) as essentially tied—0.3 points apart—yet their stated-confidence signals differ in quality, with Sonnet 5 the stronger router (AUROC 0.681 vs. 0.601; AURC 0.203 vs. 0.222). On S_{qual} at $Y = 0.76$ and $a_h = 0.90$, both qualify, but at markedly different oversight cost: Sonnet 5 meets the target while routing only 29.6% of cases to human review ($\widehat{R} = 0.856$, LCB 0.826), whereas GPT-5.4 requires 39.2% ($\widehat{R} = 0.828$, LCB 0.797)—a third more human work for the same reliability. The gap reflects routing quality, not accuracy: because Sonnet 5's

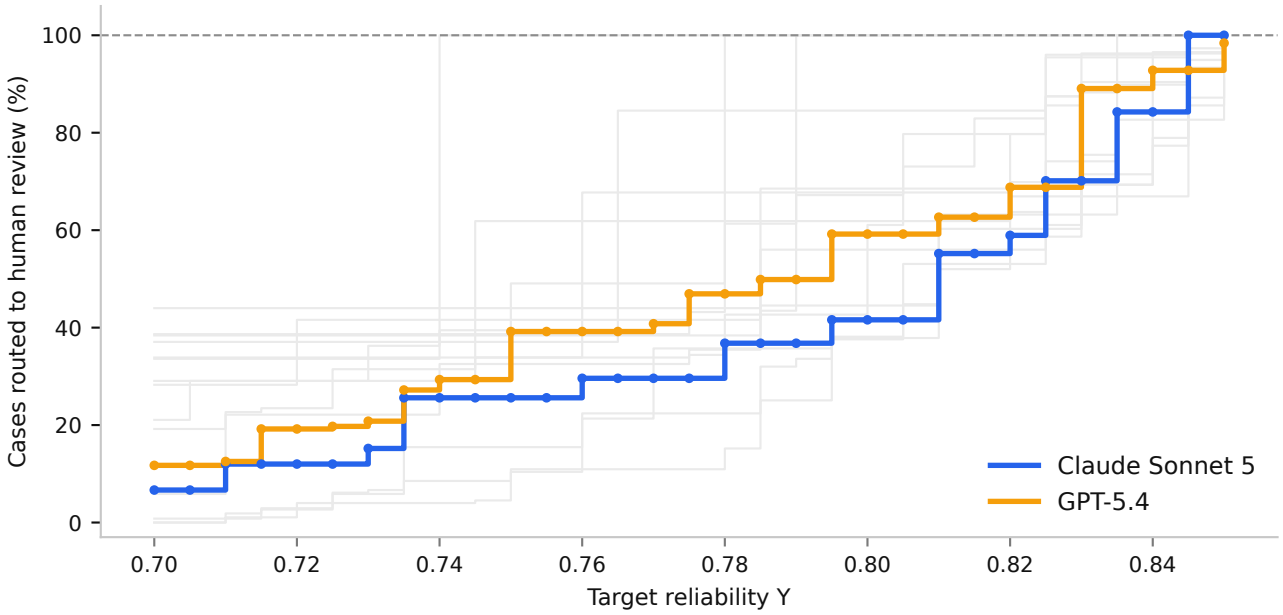

Figure 3. Reliability–oversight frontier on the held-out qualification partition, at review assumption $a_h = 0.90$. The same routing evidence is converted into the minimum human-review burden required to satisfy increasing reliability targets Y. Policies are selected on the development partition at $Y + \delta$ and frozen before qualification. Faint curves show all 16 systems; highlighted are the running-example pair: at $Y = 0.76$, Claude Sonnet 5 qualifies while routing 29.6% of cases to human review against GPT-5.4's 39.2%, despite near-identical autonomous accuracy (72.5% vs. 72.8%). Because $a_h = 0.90$ and selective success is below it at every observed coverage level, no policy can exceed reliability 0.90; every system's review burden exceeds 90% by $Y = 0.85$.

confidence more sharply separates its own successes from its failures, a less aggressive threshold already clears the reliability target and leaves more work safely autonomous.

Better routing does not translate into less review monotonically at a single target, and we report a case that shows why rather than suppress it. Qwen-3.7-Plus and MiniMax-M3 have identical development accuracy ($a_0 = 66.4$) and Qwen has the better signal on both AURC and AUROC, yet Qwen requires 61.9% review at $Y = 0.76$ against MiniMax's 33.9%. The cause is overshoot: Qwen's nearest reachable threshold delivers $\widehat{R} = 0.874$, paying for 11 points of reliability the target did not require, so single-target review burden is a noisy summary, and the frontier of Figure 3 is the fairer cross-system comparison.

Two systems can thus post near-identical benchmark scores yet support different operating policies, because one more effectively identifies the cases on which autonomous execution will fail. The difference surfaces not as another accuracy number but as the human oversight needed to reach the same reliability—the capability-versus-deployability distinction that motivates READY.

5.7 Scope and Additional Analyses

This analysis deliberately isolates terminal accept-or-escalate routing so the relationship among routing quality, reliability, and review burden stays transparent; it is not a replacement for the full CliniCARE evaluation. A richer deployment could enlarge the action space—for example, requesting additional evidence before adjudication—but doing so makes the oversight action depend on the workflow's own state and can change the subsequent trajectory, moving into the trajectory-dependent setting of

System	Review%	$\widehat{R}$	95% LCB	$a_h^{\min}$	Qualified
`Claude Code`					
Opus 5	22.4	0.839	0.807	0.690	✓
Sonnet 5	29.6	0.856	0.826	0.677	✓
`Codex`					
GPT-5.6-Sol	32.5	0.842	0.812	0.741	✓
GPT-5.6-Luna	41.6	0.796	0.764	0.890	✓
GPT-5.5	21.3	0.843	0.811	0.661	✓
GPT-5.4	39.2	0.828	0.797	0.805	✓
GPT-5.4-mini	49.1	0.850	0.822	0.773	✓
`Gemini CLI`					
Gemini-3.6-Flash	38.4	0.850	0.821	0.742	✓
Gemini-3.5-Flash	38.7	0.847	0.818	0.751	✓
Gemini-3.1-Pro	100.0	0.900	0.900	0.760	✓†
`opencode`					
DeepSeek-V4-Pro	44.0	0.839	0.810	0.787	✓
DeepSeek-V4-Flash	37.1	0.832	0.802	0.787	✓
GLM-5.2	10.9	0.762	0.725	—	–
Qwen-3.7-Plus	61.9	0.874	0.851	0.752	✓
MiniMax-M3	33.9	0.809	0.777	0.851	✓
Kimi-K2.7-Code	67.7	0.860	0.837	0.786	✓

Table 3. Held-out deployment qualification at $Y = 0.76$. One row per system, $N_{\text{qual}} = 375$, review assumption $a_h = 0.90$. Thresholds are selected on the development half at $Y + 0.05$ and frozen before qualification; the margin absorbs selection bias, without which only 46% of policies qualify across the frontier sweep, against 95% with it; at this target, 15 of 16. Following Equation 34, confidence is attached only to the observed component (exact Clopper–Pearson on accepted cases), none to the assumed a_h, so every status is conditional on that assumption. $a_h^{\min}$ (Equation 35) is the weakest review-success rate under which the policy still qualifies; — marks policies qualifying under no $a_h \in [0, 1]$. A system reviewing 100% of cases attains exactly a_h and qualifies *without deploying*, so systems are compared on Review%. †Qualifies only by routing every case to review, attaining exactly a_h; not a deployment.

Section 3.7 rather than the signal-based policy studied here.

Appendix B reports the supporting analyses: sensitivity of the frontier to the assumed review success a_h, robustness under the process-aware success definition (Equation 42), and complete per-system results. Taken together, the case study demonstrates the deployment-level information READY adds: the same task evidence used by a conventional benchmark is translated into a statistically qualified relationship among autonomous performance, routing quality, human oversight, and operating reliability.

6 Related Work

READY is not another benchmark of professional capability. It adds a deployment-qualification layer on top of such benchmarks: given evidence of how well an agent performs a workflow, it asks under what oversight policy, at what cost, and with what statistical support the agent can be deployed at a target reliability. This section situates READY against six bodies of work: capability benchmarks for knowledge and professional agents (Section 6.1); outcome, process, and evidence-based evaluation (Section 6.2); selective prediction, abstention, and learning to defer (Section 6.3); confidence, uncertainty, and calibration

Work / family	Primary question	Executable task	Interactive	Reliability	Operating cost	Human oversight	Statistical qual.
READY (ours)	Under what conditions and at what cost can an organization reliably deploy an agent on a specified workflow?	✓	✓	✓	✓	✓	✓
(§6.1) Capability and work-product benchmarks							
Humanity's Last Exam [13]	Does the model know the answer to closed-ended, frontier-difficulty questions?	×	×	×	×	×	×
Agents' Last Exam [1]	Can a generalist computer-use agent complete authentic, long-horizon professional work?	✓	×	×	×	×	×
GDPval [2]	Is the model-generated work product as good as an expert's?	✓	×	×	∼	×	×
τ^2-bench [14]	Can an agent coordinate with a user and tools to complete a task reliably across trials?	✓	✓	✓	×	×	×
CRMArena-Pro [15]	Can an agent perform professional CRM tasks across business functions and interaction types?	✓	∼	×	×	×	×
AI Agents That Matter [8]	Do agent benchmarks reward accuracy while ignoring cost?	✓	×	×	✓	×	∼
(§6.2) Process, evidence, and trajectory evaluation							
Process / trajectory eval. [16, 17]	Are the intermediate steps, tool use, and evidence correct, not just the final answer?	✓	×	×	×	×	×
(§6.3) Selective prediction, abstention, and learning to defer							
Learning to defer [18, 19]	Should the model predict this case or defer it to a human?	×	×	×	∼	✓	×
(§6.4) Confidence, uncertainty, and calibration							
Confidence / calibration [20, 21]	How likely is this output correct, and is that confidence calibrated?	×	×	∼	×	×	×
(§6.5) Cost-aware routing, cascades, and deferral economics							
Cost-aware routing & cascades [6, 7]	Can we hit a quality bar at lower cost by routing between models?	∼	×	∼	✓	×	×
(§6.6) Human oversight, scalable oversight, and assurance							
Human oversight & assurance [22, 23]	Does human review of AI improve outcomes and provide deployment assurance?	×	∼	∼	×	✓	∼

Table 4. READY as the integration point across the six related-work families of Section 6. Each row group corresponds to one subsection (§6.1–§6.6), with a representative work or method, along six axes: whether it runs an agent on *executable* multi-step work; whether it involves an *interactive* user or counterparty; whether it measures *reliability* against a target; whether it accounts for operating *cost*; whether it models *human oversight*; and whether it produces a *statistically qualified* deployment claim on held-out data. ✓ = addressed, ∼ = partial or implicit, × = not addressed. READY is the only approach that integrates all six into a single deployment-qualification claim.

as routing signals (Section 6.4); cost-aware routing, cascades, and deferral economics (Section 6.5); and human oversight, scalable oversight, and deployment assurance (Section 6.6); and, finally, the general agent-evaluation frameworks READY builds on rather than competes with (Section 6.7). READY draws on all six but integrates them into a common deployment-qualification framework that evaluates the reliability, oversight burden, and operating cost of the deployed human–AI system.

Table 4 organizes this section. Each row group corresponds to one of the families surveyed below, and the columns are the deployment dimensions READY integrates: executable and interactive task evidence, measured reliability, operating cost, human oversight, and statistical qualification. Reading across the READY row, every dimension is addressed; reading across any other row, the adjacent literatures each supply some dimensions but none combine them into a deployment-qualification claim. READY is the integration point across all six.

6.1 From Knowledge Exams to Professional-Work Agents

Evaluation of frontier models has progressed from static knowledge exams to executable professional work. Closed-ended exams such as Humanity's Last Exam [13] measure expert knowledge at the frontier of human difficulty, but they score isolated answers not an agent's ability to carry out a task or to be relied upon in operation. Agentic benchmarks instead evaluate whether a system can *do* the work. Agents' Last Exam [1] targets authentic, long-horizon computer-use workflows; GDPval [2] compares model deliverables against expert work products and estimates the associated time or cost savings; xbench [3] tracks profession-aligned productivity and technology–market fit; and recent work formalizes how to design and report benchmarks for knowledge work [24].

A large surrounding literature exercises the enterprise and tool-use skills these workflows depend on. General-assistant and web benchmarks include GAIA [25], WebArena and VisualWebArena [26, 27], and OSWorld [28]; enterprise task suites include WorkArena [29], CRMArena and CRMArena-Pro [15, 30], and Terminal-Bench [31]; conversational tool-use with a simulated user is studied by τ-bench and τ^2-bench [5, 14]; and retrieval- and browsing-heavy work is covered by BrowseComp [32] and deep-research benchmarks [33, 34]. Recent surveys further catalog this rapidly growing space [35, 36], and enterprise-grade benchmarks now span software engineering [37], simulated companies [38], workplace tasks [39, 40], and enterprise data analytics [41], building on general agent benchmarks such as AgentBench [42]. A parallel line evaluates the safety and trustworthiness of agent behavior beyond task success [43–45].

However, these evaluations primarily characterize whether the work gets done and how well, although recent work has begun to expose additional operational dimensions. Kapoor et al. [8] show that accuracy-only agent benchmarks can favor needlessly costly systems and advocate jointly reporting cost and accuracy on held-out data, while τ-bench's $pass^k$ measures whether an agent succeeds across k independent attempts and thereby captures consistency beyond single-attempt capability [5]. READY builds on these richer forms of task evidence but addresses a distinct deployment question: which oversight policy meets a prespecified reliability target at minimum operating cost, and does held-out evidence statistically support that operating point? Thus, READY consumes rather than replaces capability benchmarks, adding a reliability–oversight–cost qualification layer for deployment. The first row group of Table 4 places these capability and work-product benchmarks against READY's deployment dimensions.

6.2 Outcome, Process, and Evidence-Based Evaluation

Recent work increasingly recognizes that endpoint task success alone may provide an incomplete account of agent behavior. Process- and trajectory-oriented evaluations therefore examine intermediate execution behavior in addition to final outcomes, including whether agents follow appropriate procedures, use tools correctly, and satisfy task-specific intermediate requirements [16, 17]. Related work on factuality and grounding evaluates whether generated outputs are supported by available evidence, while recent studies further show that conclusions about hallucination or grounding can depend substantially on the evaluation protocol itself [46, 47]. Other approaches use consistency or model-internal signals to detect unsupported generation [48–50], while work on reasoning faithfulness demonstrates that plausible model explanations need not faithfully reflect the processes that produced an answer [51, 52]. More broadly, recent reviews of agent evaluation argue that deployment-oriented assessment must extend beyond aggregate task completion to consider execution quality, evidence, safety, and other workflow-specific requirements [53].

READY builds on these ideas but addresses a distinct deployment-qualification problem. It does not define a universal truthfulness or grounding metric. These properties enter through the workflow evaluator g_w when they matter for success, for example, a workflow may require evidence grounding, provenance, required tool use, or appropriate abstention alongside outcome correctness, and the success predicate ϕ_w specifies which measurements an execution must satisfy. Crucially, any process requirement used for qualification must be supported by observable execution evidence, such as retrieved sources, tool calls, or state changes, not unrecorded model reasoning. READY treats outcome and process evaluation as workflow-specific inputs to a common reliability-and-oversight methodology, not as universal properties of an agent.

6.3 Selective Prediction, Abstention, and Learning to Defer

READY's terminal accept-or-escalate setting is an instance of selective prediction, in which a system answers on a subset of cases and abstains on the remainder, trading coverage against error among accepted cases. This risk–coverage formulation has a long history: the optimal reject rule dates to Chow [54], selective classification was formalized in later work [55], and confidence-based selection was extended to deep networks [56]. The reject-option and abstention settings are surveyed by Hendrickx et al. [57] and, for language models, by Wen et al. [58], with optimal-strategy characterizations in Franc et al. [59]. A closely related literature studies *learning to defer*, in which a model chooses between predicting and passing a case to a downstream decision-maker [18, 19]. Subsequent work develops calibrated deferral [60], exact deferral algorithms [61], human-complementary predictors [62], and coverage-constrained multi-expert cooperation [63]. Distribution-free guarantees for selective decisions and thresholds are surveyed in Angelopoulos and Bates [64], Campos et al. [65].

Recent work extends these ideas to language models. AbstentionBench [66] evaluates whether models appropriately abstain on unanswerable questions; conformal methods construct coverage guarantees for LLM predictions, including settings without logit access [67, 68]; and uncertainty-aware planners request human assistance when uncertainty is high [69]. These approaches inherit the same basic tradeoff between automated coverage and error among accepted cases. Accordingly, the risk–coverage curve and its area (AURC), which we use in Section 3.4 to summarize routing quality, come directly from the selective-prediction literature [55, 56]. Likewise, the human-review pathway evaluated by READY is closely connected to the classical learning-to-defer formulation, in which responsibility is allocated between the model and a downstream decision-maker [18, 19].

READY builds on selective autonomy but changes what is optimized. An abstention or deferral rate alone does not determine whether a deployment is acceptable, because it says nothing about what happens to deferred cases. READY instead models the outcome of the declared human-review path and evaluates the reliability of the human–AI system,

$$R_{m,w}(\pi) = \mathbb{E}[u^{\pi}], \tag{46}$$

together with its expected operating cost $C_{m,w}(\pi)$, and then selects the oversight policy that meets a reliability target at minimum cost. This formulation is related to work on human-complementary prediction and coverage-constrained cooperation [62, 63], but READY's objective is deployment qualification under a workflow-level reliability requirement rather than optimization of abstention or coverage alone.

This has therefore led to three distinctions. First, human review is part of the evaluated operating policy, not merely an abstention. Second, READY separates policy selection from held-out statistical qualification, so the reported operating point is supported by independent evidence. Third, READY distinguishes *task-level abstention*—a valid workflow outcome, such as a correct indeterminate verdict in CliniCARE-Bench—from *deployment-level escalation*, the separate decision to route a completed result to human review; Appendix B develops this distinction. Thus, selective prediction and learning-to-defer methods provide the conceptual foundation for selective autonomy, while READY uses that foundation to qualify an explicit human–AI deployment operating point in terms of reliability, oversight, and cost.

6.4 Confidence, Uncertainty, and Calibration as Routing Signals

Terminal oversight requires an observable signal s_i that ranks completed executions by their likelihood of autonomous success. Many candidate signals exist. Models can self-assess correctness through elicited probabilities such as *P(True)* and *P(IK)* [20] or token-level self-evaluation [70]. Verbalized confidence [71–74], relative confidence preferences [75], semantic uncertainty [21, 73], confidence-weighted self-consistency [76], input-clarification ensembles [77], functional uncertainty [78], and logit-based methods [79] offer further ways to estimate uncertainty.

READY does not prescribe a signal; it evaluates the routing behavior induced by whatever signal the deployment policy can observe. For threshold routing, the score need not be a calibrated probability—it need only order cases so that more selective acceptance preferentially removes higher-risk executions (Section 3.4). Calibration [80–82], its behavior in language models [72, 83–85], and training-time methods for improving it [86–90] remain relevant when the numerical value of the signal is interpreted probabilistically, but calibration is not a prerequisite for qualification. Because signals that require repeated sampling or extra inference improve discrimination at a cost, READY folds that cost into the operating-cost quantity $C_{m,w}(\pi)$ instead of treating it as external to the comparison.

Surveys of confidence estimation and uncertainty quantification for language models organize the space of candidate signals [91, 92], as do hallucination surveys [93, 94]. Representative signals include semantic entropy [95], black-box confidence elicitation [96], internal-state detectors [97], and meaning-aware scores [98, 99], with large-scale comparisons of which signals are actually discriminative [100]. Selective generation [101] and conformal generation [102] apply such signals to the accept-or-defer decision directly, and when a signal is interpreted probabilistically its calibration must be measured carefully [103, 104].

6.5 Cost-Aware Routing, Cascades, and Deferral Economics

READY selects the oversight policy that meets a reliability target at minimum operating cost, which connects it to a large literature on cost-aware inference for large language models. Model *cascades* invoke a cheap model first and escalate to a more expensive one when a confidence signal is low [105–107]. FrugalGPT learns such a cascade to hit a target accuracy at reduced cost [6], AutoMix escalates under noisy self-verification via a meta-router [108], mixture-of-thought cascades defer only hard reasoning cases [109], and EcoAssistant extends the pattern to code-executing agents [110]. A parallel line *routes* each query to the cheapest adequate model: RouteLLM learns a router from preference data [7], Hybrid LLM routes by predicted difficulty [111], and Zooter routes by predicted expertise [112], with RouterBench providing a standard cost–quality evaluation [113]. Recent surveys formalize routing and cascading as a shared performance–cost optimization problem [114], a pattern that descends from classical test-time cost-sensitive cascades in detection [115].

READY shares the broader goal of reducing operating cost subject to an acceptable performance requirement, but differs in three deployment-relevant respects. First, the deferral target is a *human reviewer*, not a stronger model, so the escalation cost is human-review effort and the reliability of the deployed system depends on the review path, not on a larger model's accuracy. Second, READY replaces an average cost–quality operating point with a *statistically qualified* reliability floor established on held-out data, not a heuristic threshold or an average win rate. Third, the quality bar is a workflow-specific success predicate that may include process and policy requirements, not final-answer correctness alone. Cascade and routing methods are complementary: their confidence-based escalation logic can supply the routing signal s_i, while READY supplies the qualification procedure that turns it into a deployment guarantee.

6.6 Human Oversight, Scalable Oversight, and Deployment Assurance

READY evaluates a human–AI system under an oversight policy, connecting most directly to the work on human oversight of AI and assurance for deployment. The notion that supervision is a costly, budgeted resource originates in the scalable-oversight literature [116]; subsequent work measures whether a human assisted by an unreliable model outperforms either alone [117], studies oversight when the supervised system is stronger than its overseer [118], and proposes debate and amplification as mechanisms [119–121]. More recent work emphasizes that effective oversight requires meaningful human agency rather than nominal human presence [122, 123], while empirical studies show that human–AI teaming does not reliably outperform the stronger individual component and depends on factors such as interaction design and reviewer expertise [124, 125]. Particularly, a meta-analysis of 106 studies finds human–AI combinations on average underperform the better of human or AI alone [22], complementary performance requires more than model explanations [126], and overreliance is a central failure mode of human review [127, 128]. Mandated human oversight in particular can provide false assurance when reviewers cannot perform the checking function [129]. Finally, assurance and auditing frameworks—model cards [23], internal algorithmic auditing [130], and socio-technical safety evaluation [131]—establish reporting evaluated performance under intended deployment conditions.

READY responds to these findings directly. Rather than assuming human review improves outcomes, it treats the review path's success rate as a measured quantity or an explicit deployment assumption, and qualifies the deployed system on held-out evidence. READY consequently complements model cards and audit frameworks by reporting a statistically supported deployment profile–the reliability, oversight burden, and operating cost of a specific human–AI configuration– rather than certifying the agent in isolation

6.7 General Agent-Evaluation Frameworks

A final, infrastructure-level body of work provides the execution and harness frameworks on which agent evaluations are built, as distinct from the six methodological literatures above. Inspect AI [9] standardizes task specification, agent and tool integration, sandboxed execution, scoring, and evaluation logging; Harbor [132] packages agent evaluations as containerized tasks with verification scripts; and AgentBeats [133] provides a platform for running and comparing agent evaluations under a common protocol. These frameworks answer *how* an evaluation is executed and recorded. READY is deliberately not a competing harness: its reference implementation runs on Inspect and can consume Harbor-packaged tasks through the `inspect-harbor` adapter. READY adds the orthogonal layer of *what deployment operating point* the recorded evidence supports, selecting an oversight policy and statistically qualifying a reliability–oversight–cost profile on held-out data. In the terms of Table 4, these frameworks supply the executable-task substrate but none of the reliability, cost, oversight, or statistical-qualification dimensions; READY is the qualification layer that sits above whichever framework produced the evidence.

7 Scope, Limitations, and Future Work

READY is intended as a framework for evaluating and statistically qualifying AI-agent deployment under explicit workflow, oversight, reliability, and cost assumptions. It does not attempt to provide a universal certification of an agent or to capture every organizational consideration involved in production deployment.

Empirical scope. The empirical evaluation in this paper focuses on the trajectory-invariant, terminal accept-or-escalate setting using CliniCARE-Bench. This setting provides a transparent test of the central READY methodology: routing agent outputs between autonomous acceptance and human review to meet a target reliability at minimum oversight cost. Section 3.7 extends the formulation to trajectory-dependent policies in which review, clarification, correction, or other intervention changes subsequent execution. Evaluating such policies requires policy-in-the-loop execution or simulation; large-scale empirical validation of these settings is left to future work.

Human-oversight assumptions. READY distinguishes measured evaluation evidence from assumptions about the deployment environment. In particular, when the declared human-review path is not executed directly, its success rate, latency, and cost must be supplied as deployment assumptions. The deployment profile is conditional on those values. Future evaluations should increasingly measure reviewer effectiveness, review time, intervention type, and recovery cost directly instead of treating them as fixed parameters.

Population and distribution validity. A qualification claim applies to the workflow population represented by the evaluation and to the versioned configuration under which the evidence was collected. It does not guarantee the same reliability under arbitrary distribution shift, changes in workflow composition, environment changes, or materially different agent behavior. Continuous post-deployment monitoring is therefore important for detecting performance degradation and determining when requalification is needed, but designing such monitoring and requalification triggers is beyond the scope of this work.

Risk and deployment context. The primary READY objective represents workflow success through a workflow-specific binary event and optimizes expected operating cost subject to a reliability target. For workflows in which failures differ substantially in consequence, the framework permits additional risk constraints, including tail-risk measures.READY does not replace enterprise security, privacy, legal, regulatory, or organizational-governance processes. These considerations may define workflow constraints or evaluation criteria, but they remain outside the core deployment-qualification methodology.

Future work should extend READY along three directions: empirical evaluation of trajectory-dependent and multi-step oversight policies, direct measurement of human-review effectiveness and economics, and broader validation across enterprise workflows and domains. The open interface described in Section 4 supports these extensions without changing the underlying qualification objective.

8 Conclusion

Professional-work benchmarks measure how well an AI agent can perform a task autonomously. Enterprise deployment requires a different determination: whether the agent can be deployed at a required level of reliability, with an acceptable amount of human oversight and operating cost.

Reliable Enterprise Agent Deployment (READY) formulates this as a deployment-qualification problem. For a given workflow, READY evaluates agent execution, selects an oversight policy that satisfies a target reliability at minimum operating cost, and statistically qualifies that operating point on held-out evidence. A deployment profile is not an intrinsic score of the agent. It is a conditional statement about how that agent can be deployed under a particular workflow, oversight model, and set of operating assumptions.

READY separates workflow-specific definitions of successful work from a common qualification methodology and provides an open runtime testbed for executing and extending such evaluations. Trajectory-invariant policies can be evaluated efficiently from saved execution evidence, while trajectory-dependent policies use the same optimization objective but require policy-in-the-loop execution or simulation. CliniCARE-Bench provides an end-to-end instantiation of this framework and illustrates how autonomous task performance, routing quality, human oversight, and deployment reliability can be analyzed together.

As agent capabilities improve, the relevant question is not simply whether benchmark scores rise, but how much human oversight remains necessary to meet a specified reliability requirement. READY provides a framework for measuring that transition and for comparing systems by the cost of reliable deployment, not autonomous performance alone.

Author Contributions

Y. Xue conceived the project, defined the vision and direction, led the manuscript effort, and supervised the work. V. Chatrath led benchmark execution, including the Contributor network and experiments, and served as primary manuscript lead. B. Zhu and J. Fan served as technical and manuscript leads and were primary developers. G. Pu was a primary developer. S. D. Tiwari, S. Dan, and R. Young developed the benchmark pipeline. A. Shanker contributed the mathematical formulation. Y. Li, Y. Yao, and M. Yang contributed to manuscript iteration. D. Y. Zhang, Y. He, Y. Liu, C. Wang, and Z. Yin served as research advisors. All authors contributed to the writing, reviewed and approved the final manuscript.

References

[1] Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, et al. Agents' Last Exam, 2026. URL https://arxiv.org/abs/2606.05405.

[2] Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI model performance on real-world economically valuable tasks, 2025. URL https://arxiv.org/abs/2510.04374.

[3] Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, et al. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations, 2025. URL https://arxiv.org/abs/2506.13651.

[4] Veronica Chatrath, Bryan Zhu, George Pu, Jingxuan Fan, Apaar Shanker, Varun Ursekar, Anahita Sharma, Jason Qin, Keqi Han, Soham Dinesh Tiwari, Soham Dan, Vijay Kalmath, Yuan Li, Daniel Yue Zhang, Chenguang Wang, Zainab Doctor, Zhijun Yin, Nigam H. Shah, and Yuan Xue. CliniCARE-Bench: Clinical calibrated audit of medical reasoning in EHR. *arXiv preprint arXiv:2608.07796*, 2026. URL https://arxiv.org/abs/2608.07796.

[5] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

[6] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023.

[7] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pages 34433–34448, 2025.

[8] Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *arXiv preprint arXiv:2407.01502*, 2024.

[9] UK AI Security Institute. Inspect ai: Framework for large language model evaluations, May 2024. URL https://github.com/UKGovernmentBEIS/inspect_ai. Released May 2024.

[10] Kidney Disease: Improving Global Outcomes (KDIGO) Acute Kidney Injury Work Group. KDIGO clinical practice guideline for acute kidney injury. *Kidney International Supplements*, 2012.

[11] Satyajith Amaran, Nikolaos V. Sahinidis, Bikram Sharda, and Scott J. Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, 2016. doi: 10.1007/s10479-015-2019-x.

[12] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. Springer, 2017. doi: 10.1007/978-3-319-68913-5.

[13] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, Summer Yue, Alexandr Wang, Dan Hendrycks, et al. Humanity's Last Exam, 2025. URL https://arxiv.org/abs/2501.14249.

[14] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

[15] Kung-Hsiang Huang, Akshara Prabhakar, Onkar Thorat, Divyansh Agarwal, Prafulla Kumar Choubey, Yixin Mao, Silvio Savarese, Caiming Xiong, and Chien-Sheng Wu. CRMArena-Pro: Holistic assessment of LLM agents across diverse business scenarios and interactions. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=EPlpe3Fx1x.

[16] Haochen Sun, Shuwen Zhang, Lujie Niu, Lei Ren, Hao Xu, Hao Fu, Fangkun Zhao, Caixia Yuan, and Xiaojie Wang. Collab-overcooked: Benchmarking and evaluating large language models as collaborative agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4922–4951, 2025.

[17] Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. Mlr-bench: Evaluating ai agents on open-ended machine learning research. *Advances in Neural Information Processing Systems*, 38, 2026.

[18] David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/09d37c08f7b129e96277388757530c72-Abstract.html.

[19] Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 7076–7087. PMLR, 2020. URL https://proceedings.mlr.press/v119/mozannar20b.html.

[20] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

[21] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

[22] Michelle Vaccaro, Abdullah Almaatouq, and Thomas W. Malone. When combinations of humans and ai are useful: A systematic review and meta-analysis. *Nature Human Behaviour*, 8:2293–2303, 2024. doi: 10.1038/s41562-024-02024-1.

[23] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, pages 220–229, 2019. doi: 10.1145/3287560.3287596.

[24] Yining Hua, Hongbin Na, Cyrus Ayubcha, and Levi Lian. Design and report benchmarks for knowledge work, 2026. URL https://arxiv.org/abs/2605.23262.

[25] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

[26] 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. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

[27] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pages 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

[28] 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. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

[29] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

[30] Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng Wu. CRMArena: Understanding the capacity of LLM agents to perform professional CRM tasks in realistic environments. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2025*, pages 3830–3850, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.194. URL https://aclanthology.org/2025.naacl-long.194/.

[31] 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.

[32] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

[33] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. DeepResearch Bench: A comprehensive benchmark for deep research agents, 2025. URL https://arxiv.org/abs/2506.11763.

[34] Amirhossein Abaskohi, Tianyi Chen, Miguel Mu noz Mármol, Curtis Fox, Amrutha Varshini Ramesh, Étienne Marcotte, et al. DRBench: A realistic benchmark for enterprise deep research, 2025. URL https://arxiv.org/abs/2510.00172.

[35] Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of llm-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

[36] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. *arXiv preprint arXiv:2507.21504*, 2025.

[37] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations (ICLR)*, 2024.

[38] Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024.

[39] Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. Workbench: a benchmark dataset for agents in a realistic workplace setting. *arXiv preprint arXiv:2405.00823*, 2024.

[40] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[41] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. In *International Conference on Learning Representations (ICLR)*, 2025.

[42] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang.

Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

[43] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, J. Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=AC5n7xHuR1.

[44] Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, Nir Mashkif, and Segev Shlomov. ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. In *The Fourteenth International Conference on Learning Representations, ICLR 2026*. OpenReview.net, 2026. URL https://openreview.net/forum?id=MuCDzH0ctf.

[45] Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents, 2024. URL https://arxiv.org/abs/2412.14470.

[46] Gregor Geigle, Radu Timofte, and Goran Glavaš. Does object grounding really reduce hallucination of large vision-language models? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2728–2742, 2024.

[47] Denis Janiak, Jakub Binkowski, Albert Sawczyn, Bogdan Gabrys, Ravid Shwartz-Ziv, and Tomasz Jan Kajdanowicz. The illusion of progress: Re-evaluating hallucination detection in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34728–34745, 2025.

[48] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9004–9017, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

[49] Jakub Binkowski, Denis Janiak, Albert Sawczyn, Bogdan Gabrys, and Tomasz Jan Kajdanowicz. Hallucination detection in llms using spectral features of attention maps. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24354–24385, 2025.

[50] Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=KRnsX5Em3W.

[51] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner,

Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

[52] Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/ed3fea9033a80fea1376299fa7863f4a-Abstract-Conference.html.

[53] Tanzila Kehkashan, Muhammad Abdullah, Ahmad Sami Al-Shamayleh, Nikola Ivković, Nor Azman Ismail, Sharifah Sakinah Syed Ahmad, Abdul Rehman, and Adnan Akhunzada. From benchmarks to deployment: a comprehensive review of agentic ai evaluation. *Artificial intelligence review*, 2026.

[54] C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970. doi: 10.1109/TIT.1970.1054406.

[55] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010. URL https://jmlr.org/papers/v11/el-yaniv10a.html.

[56] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 2017. URL https://arxiv.org/abs/1705.08500.

[57] Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. Machine learning with a reject option: A survey. *Machine Learning*, 113:3073–3110, 2024.

[58] Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics (TACL)*, 13:529–556, 2025.

[59] Vojtěch Franc, Daniel Průša, and Václav Voráček. Optimal strategies for reject option classifiers. *Journal of Machine Learning Research*, 24(11):1–49, 2023.

[60] Rajeev Verma and Eric Nalisnick. Calibrated learning to defer with one-vs-all classifiers. In *International Conference on Machine Learning (ICML)*, 2022.

[61] Hussein Mozannar, Hunter Lang, Dennis Wei, Prasanna Sattigeri, Subhro Das, and David Sontag. Who should predict? exact algorithms for learning to defer to humans. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

[62] Mohammad-Amin Charusaie, Hussein Mozannar, David Sontag, and Samira Samadi. Sample efficient learning of predictors that complement humans. In *International Conference on Machine Learning (ICML)*, 2022.

[63] Zheng Zhang, Cuong Nguyen, Kevin Wells, Thanh-Toan Do, David Rosewarne, and Gustavo Carneiro. Coverage-constrained human-ai cooperation with multiple experts. *arXiv preprint arXiv:2411.11976*, 2024.

[64] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[65] Margarida M. Campos, António Farinhas, Chrysoula Zerva, Mário A. T. Figueiredo, and André F. T. Martins. Conformal prediction for natural language processing: A survey. *Transactions of the Association for Computational Linguistics (TACL)*, 12:1497–1516, 2024.

[66] Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J. Bell. Abstentionbench: Reasoning LLMs fail on unanswerable questions. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla, editors, *Advances in Neural Information Processing Systems 38 (NeurIPS 2025), Datasets and Benchmarks Track*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/fb122bfc3f0127a94ded048b5b03496f-Abstract-Datasets_and_Benchmarks_Track.html.

[67] Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. Conformal prediction with large language models for multi-choice question answering, 2023. URL https://arxiv.org/abs/2305.18404.

[68] Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. API is enough: Conformal prediction for large language models without logit-access. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 979–995, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.54. URL https://aclanthology.org/2024.findings-emnlp.54/.

[69] Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pages 661–682. PMLR, 2023. URL https://proceedings.mlr.press/v229/ren23a.html.

[70] Jie Ren, Yao Zhao, Tu Vu, Peter J. Liu, and Balaji Lakshminarayanan. Self-evaluation improves selective generation in large language models. In Javier Antorán, Arno Blaas, Kelly Buchanan, Fan Feng, Vincent Fortuin, Sahra Ghalebikesabi, Andreas Kriegler, Ian Mason, David Rohde, Francisco J. R. Ruiz, Tobias Uelwer, Yubin Xie, and Rui Yang, editors, *Proceedings on "I Can't Believe It's Not Better: Failure Modes in the Age of Foundation Models" at NeurIPS 2023 Workshops*, volume 239 of *Proceedings of Machine Learning Research*, pages 49–64. PMLR, 2023.

[71] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=8s8K2UZGTZ.

[72] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5433–5442, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.330. URL https://aclanthology.org/2023.emnlp-main.330/.

[73] Ziwei Ji, Lei Yu, Yeskendir Koishekenov, Yejin Bang, Anthony Hartshorn, Alan Schelten, Cheng Zhang, Pascale Fung, and Nicola Cancedda. Calibrating verbal uncertainty as a linear feature to reduce hallucinations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3769–3793, 2025.

[74] Gabrielle Kaili-May Liu, Gal Yona, Avi Caciularu, Idan Szpektor, Tim GJ Rudner, and Arman Cohan. Metafaith: Faithful natural language uncertainty expression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29600–29644, 2025.

[75] Vaishnavi Shrivastava, Ananya Kumar, and Percy Liang. Language models prefer what they know: Relative confidence estimation via confidence preferences, 2025. URL https://arxiv.org/abs/2502.01126.

[76] Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20090–20111, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1030. URL https://aclanthology.org/2025.findings-acl.1030/.

[77] Bairu Hou, Yujian Liu, Kaizhi Qian, Jacob Andreas, Shiyu Chang, and Yang Zhang. Decomposing uncertainty for large language models through input clarification ensembling. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 19023–19042. PMLR, 2024.

[78] Ruijia Niu, Dongxia Wu, Rose Yu, and Yi-An Ma. Functional-level uncertainty quantification for calibrated fine-tuning on llms, 2024. URL https://arxiv.org/abs/2410.06431.

[79] Mingyu Derek Ma, Yanna Ding, Zijie Huang, Jianxi Gao, Yizhou Sun, and Wei Wang. Inferring from logits: Exploring best practices for decoding-free generative candidate selection. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*, pages 33125–33144. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1589. URL https://aclanthology.org/2025.acl-long.1589/.

[80] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017. URL http://proceedings.mlr.press/v70/guo17a.html.

[81] Johnathan Xie, Annie S. Chen, Yoonho Lee, Eric Mitchell, and Chelsea Finn. Calibrating language models with adaptive temperature scaling. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 18128–18138, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1007. URL https://aclanthology.org/2024.emnlp-main.1007/.

[82] Yukun Li, Sijia Wang, Lifu Huang, and Liping Liu. Graph-based confidence calibration for large language models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=BDPvuD5FTg.

[83] Mark Braverman, Xinyi Chen, Sham M. Kakade, Karthik Narasimhan, Cyril Zhang, and Yi Zhang. Calibration, entropy rates, and memory in language models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 1089–1099. PMLR, 2020. URL http://proceedings.mlr.press/v119/braverman20a.html.

[84] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407. URL https://aclanthology.org/2021.tacl-1.57/.

[85] Yangyi Chen, Lifan Yuan, Ganqu Cui, Zhiyuan Liu, and Heng Ji. A close look into the calibration of pre-trained language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 1343–1367, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.75. URL https://aclanthology.org/2023.acl-long.75/.

[86] Sanyam Kapoor, Nate Gruver, Manley Roberts, Arka Pal, Samuel Dooley, Micah Goldblum, and Andrew Gordon Wilson. Calibration-tuning: Teaching large language models to know what they don't know. In Raúl Vázquez, Hande Celikkanat, Dennis Ulmer, Jörg Tiedemann, Swabha Swayamdipta, Wilker Aziz, Barbara Plank, Joris Baan, and Marie-Catherine de Marneffe, editors, *Proceedings of the 1st Workshop on Uncertainty-Aware NLP (UncertaiNLP 2024)*, pages 1–14, St Julians, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.uncertainlp-1.1. URL https://aclanthology.org/2024.uncertainlp-1.1/.

[87] Elias Stengel-Eskin, Peter Hase, and Mohit Bansal. LACIE: Listener-aware finetuning for calibration in large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, volume 37, pages 43080–43106. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4b8eaf3bcdc105423a972ed90eb07217-Abstract-Conference.html.

[88] Yao Zhao, Misha Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J. Liu. Calibrating sequence likelihood improves conditional language generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=0qSOodKmJaN.

[89] Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. SLiC-HF: Sequence likelihood calibration with human feedback, 2023. URL https://arxiv.org/abs/2305.10425.

[90] Zhengyan Shi, Sander Land, Acyr Locatelli, Matthieu Geist, and Max Bartolo. Understanding likelihood over-optimisation in direct alignment algorithms, 2024. URL https://arxiv.org/abs/2410.11677.

[91] Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

[92] Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *arXiv preprint arXiv:2412.05563*, 2024.

[93] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Delong Chen, Wenliang Dai, Ho Shu Chan, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023. doi: 10.1145/3571730.

[94] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[95] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. doi: 10.1038/s41586-024-07421-0.

[96] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *International Conference on Learning Representations (ICLR)*, 2024.

[97] Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. Inside: Llms' internal states retain the power of hallucination detection. In *International Conference on Learning Representations (ICLR)*, 2024.

[98] Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[99] Yavuz Faruk Bakman, Duygu Nur Yaldiz, Baturalp Buyukates, Chenyang Tao, Dimitrios Dimitriadis, and Salman Avestimehr. Mars: Meaning-aware response scoring for uncertainty estimation in generative llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[100] Roman Vashurin, Ekaterina Fadeeva, Artem Vazhentsev, Lyudmila Rvanova, Akim Tsvigun, Daniil Vasilev, Rui Xing, Abdelrahman Boda Sadallah, Kirill Grishchenkov, Sergey Petrakov, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, Maxim Panov, and Artem Shelmanov. Benchmarking uncertainty quantification methods for large language models with lm-polygraph. *Transactions of the Association for Computational Linguistics (TACL)*, 13:220–248, 2025.

[101] Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J. Liu. Out-of-distribution detection and selective generation for conditional language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[102] Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling. In *International Conference on Learning Representations (ICLR)*, 2024.

[103] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pages 2901–2907, 2015. doi: 10.1609/aaai.v29i1.9602.

[104] Jeremy Nixon, Michael W. Dusenberry, Ghassen Jerfel, Timothy Nguyen, Jeremiah Liu, Linchuan Zhang, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[105] António Farinhas, Nuno M Guerreiro, Sweta Agrawal, Ricardo Rei, and André FT Martins. Translate smart, not hard: Cascaded translation systems with quality-aware deferral. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26730–26744, 2025.

[106] Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 12987–13010. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/dekoninck25a.html.

[107] Yuanzhe Shen, Yide Liu, Zisu Huang, Ruicheng Yin, Xiaoqing Zheng, and Xuan-Jing Huang. Sater: A self-aware and token-efficient approach to routing and cascading. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10526–10540, 2025.

[108] Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[109] Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

[110] Jieyu Zhang, Ranjay Krishna, Ahmed H. Awadallah, and Chi Wang. Ecoassistant: Using llm assistant more affordably and accurately, 2023.

[111] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V.S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations (ICLR)*, 2024.

[112] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2024.

[113] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system, 2024.

[114] Clovis Varangot-Reille, Christophe Bouvard, Antoine Gourru, Mathieu Ciancone, Marion Schaeffer, and François Jacquenet. Doing more with less: A survey on routing strategies for resource

optimisation in large language model-based systems. *Journal of Artificial Intelligence Research (JAIR)*, 2025. doi: 10.1613/jair.1.19801.

[115] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511–I–518, 2001. doi: 10.1109/CVPR.2001.990517.

[116] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[117] Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

[118] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[119] Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

[120] Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.

[121] Hao Lang, Fei Huang, and Yongbin Li. Debate helps weak-to-strong generalization. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, pages 27410–27418, 2025. doi: 10.1609/aaai.v39i26.34952.

[122] Liming Zhu, Qinghua Lu, Ming Ding, Sung Une Lee, and Chen Wang. Designing meaningful human oversight in ai. *AI and Ethics*, 6(3):286, 2026.

[123] Andreas Tsamados, Luciano Floridi, and Mariarosaria Taddeo. Human control of ai systems: from supervision to teaming. *AI and Ethics*, 5(2):1535–1548, 2025.

[124] Jingyu Tang, Chaoran Chen, Jiawen Li, Zhiping Zhang, Bingcan Guo, Ibrahim Khalilov, Simret Araya Gebreegziabher, Bingsheng Yao, Dakuo Wang, Yanfang Ye, et al. Dark patterns meet gui agents: Llm agent susceptibility to manipulative interfaces and the role of human oversight. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, pages 1–26, 2026.

[125] Peng Liu, Jiaxin Zhang, Shuaiqi Chen, and Shanguang Chen. Human-ai teaming in healthcare: 1+ 1> 2? *npj Artificial Intelligence*, 1(1):47, 2025.

[126] Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel S. Weld. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI)*, 2021.

[127] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z. Gajos. To trust or to think: Cognitive forcing functions can reduce overreliance on ai in ai-assisted decision-making. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–21, 2021. doi: 10.1145/3449287.

[128] Vivian Lai, Chacha Chen, Q. Vera Liao, Alison Smith-Renner, and Chenhao Tan. Towards a science of human-ai decision making: A survey of empirical studies. *arXiv preprint arXiv:2112.11471*, 2021.

[129] Ben Green. The flaws of policies requiring human oversight of government algorithms. *Computer Law & Security Review*, 45:105681, 2022. doi: 10.1016/j.clsr.2022.105681.

[130] Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT*)*, pages 33–44, 2020. doi: 10.1145/3351095.3372873.

[131] Laura Weidinger, Maribeth Rauh, Nahema Marchal, Arianna Manzini, Lisa Anne Hendricks, Juan Mateos-Garcia, Stevie Bergman, Jackie Kay, Conor Griffin, Ben Bariach, Iason Gabriel, Verena Rieser, and William Isaac. Sociotechnical safety evaluation of generative ai systems. *arXiv preprint arXiv:2310.11986*, 2023.

[132] Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, 2026. URL https://doi.org/10.5281/zenodo.20953922.

[133] Xiaoyuan Liu, Jianhong Tu, Yuqi Chen, Siyuan Xie, Sihan Ren, Tianneng Shi, Gal Gantar, Evan Sandoval, Donghyun Lee, Daniel Miao, et al. Agentbeats: Agentifying agent assessment for openness, standardization, and reproducibility. *arXiv preprint arXiv:2606.13608*, 2026.

A Formal definitions

A.1 Threshold Routing and Monotonicity of the Routing Signal

Consider the trajectory-invariant terminal-oversight setting. For task instance i, autonomous execution produces an observable routing signal $s_i \in \mathbb{R}$ and an evaluation outcome $u_i \in \{0, 1\}$, where $u_i = 1$ denotes successful autonomous execution. The outcome u_i is available to the evaluator during policy development and qualification, but is generally not known when the routing decision is made.

A threshold policy π_τ accepts the agent output autonomously when $s_i \geq \tau$ and routes it to human review otherwise:

$$\pi_\tau(s_i) = \begin{cases} \text{accept}, & s_i \geq \tau, \\ \text{review}, & s_i < \tau. \end{cases} \tag{47}$$

For threshold τ, define the autonomous coverage

$$c(\tau) = \Pr(s \geq \tau), \tag{48}$$

and the success probability among autonomously accepted cases,

$$a(\tau) = \Pr(u = 1 \mid s \geq \tau). \tag{49}$$

We call s a *monotone routing signal* if increasing the threshold does not decrease the success probability of the retained autonomous cases:

$$\tau_1 < \tau_2 \quad \Longrightarrow \quad a(\tau_1) \leq a(\tau_2), \tag{50}$$

for thresholds with nonzero autonomous coverage. Equivalently, increasingly selective autonomous acceptance should preferentially remove higher-risk cases.

A stronger sufficient condition is pointwise monotonicity,

$$s_1 < s_2 \quad \Longrightarrow \quad \Pr(u = 1 \mid s = s_1) \leq \Pr(u = 1 \mid s = s_2), \tag{51}$$

but READY does not require this stronger condition. In finite evaluation samples, local departures from monotonicity may occur; what matters operationally is whether the signal induces a sufficiently ordered risk–coverage relationship to support reliable selective routing.

Importantly, monotonicity is distinct from calibration. A routing signal may provide a useful ordering of cases even when s_i is not numerically equal to the probability of successful execution. Calibration is required only when the numerical value of the signal is interpreted probabilistically; threshold-based routing principally requires discrimination and ordering.

A.2 Tail-Risk Estimation for Risk-Sensitive Qualification

The main text treats workflow-specific risk as an optional extension to the common READY reliability constraint. This appendix gives one concrete implementation using conditional value-at-risk (CVaR).

The construction is not required for workflows in which binary deployment success adequately captures the consequences of failure.

Let

$$L_i^\pi = L_w(Z_i^\pi) \geq 0 \tag{52}$$

denote the workflow-specific loss incurred on qualification instance i under a fixed oversight policy π. The loss function is defined as part of the workflow specification and may encode, for example, the severity of a failed action, downstream recovery cost, or another consequence not represented adequately by the binary success indicator u_i^π.

For confidence level $\alpha \in (0,1)$, the value-at-risk is

$$\mathrm{VaR}_\alpha(L^\pi) = \inf\{\ell : \Pr(L^\pi \leq \ell) \geq \alpha\}. \tag{53}$$

CVaR measures the expected loss in the upper tail beyond this quantile. For estimation, we use the equivalent optimization representation

$$\mathrm{CVaR}_\alpha(L^\pi) = \min_{\eta \in \mathbb{R}} \left[\eta + \frac{1}{1-\alpha}\mathbb{E}\left[(L^\pi - \eta)_+\right]\right], \tag{54}$$

where $(x)_+ = \max(x, 0)$. This form remains well defined for discrete empirical loss distributions and avoids requiring a separate estimator of the tail quantile.

Held-out estimator. After policy selection, the policy $\widehat{\pi}$ and the risk specification (L_w, α, B_w) are frozen before evaluation on the qualification set. Given losses

$$L_1^{\widehat{\pi}}, \ldots, L_N^{\widehat{\pi}}, \tag{55}$$

the empirical CVaR estimator is

$$\widehat{\mathrm{CVaR}}_\alpha = \min_{\eta \in \mathbb{R}} \left[\eta + \frac{1}{(1-\alpha)N}\sum_{i=1}^{N}\left(L_i^{\widehat{\pi}} - \eta\right)_+\right]. \tag{56}$$

For trajectory-invariant policies, the losses may be computed by replaying the frozen policy over saved qualification trajectories. For trajectory-dependent policies, the losses must be obtained from qualification executions or simulations performed under $\widehat{\pi}$ itself.

Statistical qualification. A point estimate of CVaR is not sufficient to establish a deployment-risk constraint. READY therefore constructs a one-sided upper confidence bound

$$\mathrm{CVaR}_{\mathrm{UCB}}^{1-\delta}(\widehat{\pi}) \tag{57}$$

at confidence level $1-\delta$ and qualifies the policy with respect to the tail-risk constraint only if

$$\mathrm{CVaR}_{\mathrm{UCB}}^{1-\delta}(\widehat{\pi}) \leq B_w. \tag{58}$$

In the reference procedure, this upper bound is obtained by bootstrap resampling the held-out qualification set. For each bootstrap replicate $b = 1, \ldots, B$, READY resamples qualification instances, recomputes $\widehat{\mathrm{CVaR}}_{\alpha}^{(b)}$, and uses the appropriate upper percentile of the resulting bootstrap distribution as the one-sided confidence bound.

The bootstrap unit must match the unit of statistical generalization. If multiple stochastic runs are collected for the same task instance, all runs from that instance are resampled together. Likewise, if observations are clustered by patient, customer, account, or another natural unit, resampling is performed at that cluster level rather than treating individual observations as independent.

Uncertain downstream outcomes. If L_i^{π} depends on an outcome that is not directly observed in the evaluation, for example, the success or cost of a hypothetical human-review path, the corresponding uncertainty must be propagated into the risk estimate. READY may do this by jointly sampling the uncertain review parameters and the qualification instances in each bootstrap replicate. A risk quantity computed by substituting a single assumed review-success rate or recovery cost should therefore be reported as conditional on that assumption rather than as an empirically qualified tail-risk estimate.

Pre-specification and tail support. The loss function L_w, tail level α, risk tolerance B_w, and confidence level $1 - \delta$ must be specified before examining the held-out qualification results. They must not be tuned to make a candidate policy pass qualification.

Tail-risk estimates can be statistically weak when the qualification set contains few observations in the relevant tail. For example, an $\alpha = 0.99$ CVaR estimate is informed primarily by roughly the worst one percent of the sampled outcomes. READY therefore reports the effective number of observations contributing to the estimated tail together with the point estimate and confidence bound. If the available qualification sample does not support a sufficiently precise upper bound, the appropriate result is *insufficient evidence for risk qualification*, rather than a relaxed risk threshold.

B Supplementary Analyses for CliniCARE-Bench

This appendix supplements the four-way deployment-qualification analysis of Section 5 with the held-out lower-bound computation, sensitivity checks, and complete per-system results. All policies follow the global-routing definition and margin-adjusted threshold selection of Section 5.2; all thresholds are selected on the development partition and frozen before evaluation on the held-out qualification partition.

B.1 Reproducing the Held-Out Lower Bound

The 95% LCB column of Table 3 instantiates Equation 34, attaching confidence only to the observed accepted component. Fix the threshold $\widehat{\tau}$ selected on S_{dev} at $Y + \delta$ ($\delta = 0.05$; Equation 45) and let $N_{\mathrm{qual}} = 375$.

1. **Route.** For each qualification case set $d_i = \mathbf{1}\{s_i \geq \widehat{\tau}\}$ (Equation 26). Let $n_{\mathrm{acc}} = \sum_i d_i$ be the number of accepted cases and $k_{\mathrm{acc}} = \sum_i d_i u_i$ the number of those that are autonomously successful, so the

reviewed fraction is $(N_{\text{qual}} - n_{\text{acc}})/N_{\text{qual}}$ and $\widehat{p}_{\text{acc}} = k_{\text{acc}}/n_{\text{acc}}$.

2. **Point estimate.** Combine the observed accepted rate with the assumed review success via Equation 32: $\widehat{R} = \frac{n_{\text{acc}}}{N_{\text{qual}}}\widehat{p}_{\text{acc}} + \frac{N_{\text{qual}}-n_{\text{acc}}}{N_{\text{qual}}}\, a_h$.
3. **Lower bound.** Compute the one-sided 95% exact Clopper–Pearson lower bound $\underline{p}_{\text{acc}}$ for the accepted-case success probability from $(k_{\text{acc}}, n_{\text{acc}})$, i.e. the $\alpha = 0.05$ quantile of the Beta$(k_{\text{acc}},\, n_{\text{acc}} - k_{\text{acc}} + 1)$ distribution (with $\underline{p}_{\text{acc}} = 0$ if $k_{\text{acc}} = 0$). No confidence is attached to the assumed term, so

$$95\%\ \text{LCB} = \frac{n_{\text{acc}}}{N_{\text{qual}}}\,\underline{p}_{\text{acc}} + \frac{N_{\text{qual}} - n_{\text{acc}}}{N_{\text{qual}}}\, a_h.$$

4. **Qualify.** The policy qualifies iff 95% LCB $\geq Y$.
5. **Break-even.** $a_h^{\min}$ (Equation 35) is the smallest $a_h \in [0,1]$ for which the LCB expression reaches Y; because the LCB is nondecreasing in a_h this is a one-dimensional solve, and — marks policies for which no $a_h \in [0,1]$ suffices.

B.2 Sensitivity to the Human-Review Assumption

The main analysis treats human-review success a_h as a declared deployment assumption (Equation 43). Here we recompute the reliability–oversight frontier and the qualification outcomes of Section 5.4 under alternative values of a_h, reporting how the required review burden and qualified/not-qualified status change with the assumed quality of the human review path. We repeat the entire dev-select/freeze/qualify procedure at $a_h \in \{0.80, 0.85, 0.90, 0.95, 1.00\}$ over the same 31-target sweep, giving $31 \times 16 = 496$ policies per assumption. Nothing about the execution evidence changes; only the declared assumption does.

Table 5 reports the per-system consequences at the representative target $Y = 0.76$. The aggregate picture over the full sweep is that the review assumption, not agent capability, governs how much of the reliability range is reachable at all:

a_h	Qualified	Mean Review%	Highest Y reached by a *deploying* policy
0.80	318/496	76.3	0.775
0.85	472/496	65.1	0.805
0.90	469/496	53.0	≥ 0.850
0.95	467/496	44.6	≥ 0.850
1.00	463/496	38.9	≥ 0.850

Weakening the assumed review path from $a_h = 0.90$ to 0.80 raises mean review burden by 23 percentage points and caps the highest reliability any deploying policy attains at 0.775; at that assumption 311 of 496 policies route every case to review, qualifying only by not deploying. The last column is censored from above at $a_h \geq 0.90$ because the sweep ends at $Y = 0.85$, so those entries are lower bounds rather than ceilings.

Two features of the table deserve comment because they are properties of the qualification procedure rather than of the systems. First, the number of qualifying policies is *not* monotone in a_h (472 at $a_h = 0.85$ against 469 at $a_h = 0.90$). A more generous review assumption relaxes the development-set constraint, which lets the optimization select a lower threshold and accept more cases; the resulting accepted set is

larger but less reliable, and can fail the lower-confidence-bound test on the held-out partition that the more conservative policy passed. GPT-5.6-Luna at $Y = 0.76$ is the clean instance: it qualifies at $a_h = 0.80$ and at $a_h = 0.90$, but not at $a_h = 0.85$. This is the expected behaviour of selecting on one partition and testing on another, and it is a reason to read the frontier rather than any single cell. Second, review burden is weakly decreasing in a_h for every system, but in steps, because each system can only move between the thresholds its stated confidence actually reaches. Gemini-3.1-Pro is unchanged at 100% review for every $a_h \leq 0.95$ and only drops to 33.6% at $a_h = 1.00$, which reflects the granularity limit of Section 5.3, not a response to the assumption.

	$a_h = 0.80$		$a_h = 0.85$		$a_h = 0.90$		$a_h = 0.95$		$a_h = 1.00$	
System	Rev%	Q	Rev%	Q	Rev%	Q	Rev%	Q	Rev%	Q
`Claude Code`										
Opus 5	22.4	✓	22.4	✓	22.4	✓	22.4	✓	10.4	✓
Sonnet 5	41.6	✓	29.6	✓	29.6	✓	25.6	✓	25.6	✓
`Codex`										
GPT-5.6-Sol	44.5	✓	35.5	✓	32.5	✓	32.5	✓	32.5	✓
GPT-5.6-Luna	56.0	✓	41.6	–	41.6	✓	41.6	✓	41.6	✓
GPT-5.5	35.7	✓	35.7	✓	21.3	✓	15.5	✓	15.5	✓
GPT-5.4	68.8	✓	49.9	✓	39.2	✓	29.3	✓	27.2	✓
GPT-5.4-mini	100.0	✓	67.2	✓	49.1	✓	39.5	✓	36.3	✓
`Gemini CLI`										
Gemini-3.6-Flash	100.0	✓	38.4	✓	38.4	✓	38.4	✓	38.4	✓
Gemini-3.5-Flash	100.0	✓	100.0	✓	38.7	✓	38.7	✓	38.7	✓
Gemini-3.1-Pro	100.0	✓	100.0	✓	100.0	✓	100.0	✓	33.6	✓
`opencode`										
DeepSeek-V4-Pro	100.0	✓	44.0	✓	44.0	✓	44.0	✓	44.0	✓
DeepSeek-V4-Flash	100.0	✓	84.5	✓	37.1	✓	37.1	✓	37.1	✓
GLM-5.2	37.6	✓	10.9	–	10.9	–	10.9	–	10.9	–
Qwen-3.7-Plus	100.0	✓	61.9	✓	61.9	✓	61.9	✓	31.5	✓
MiniMax-M3	100.0	✓	42.7	✓	33.9	✓	33.9	✓	33.9	✓
Kimi-K2.7-Code	100.0	✓	67.7	✓	67.7	✓	33.9	✓	29.1	✓

Table 5. Held-out qualification at $Y = 0.76$ under alternative human-review assumptions. $N_{\text{qual}} = 375$; thresholds reselected on S_{dev} at $Y + 0.05$ for each a_h and frozen. Rev% is the fraction of qualification cases routed to human review; Q marks whether the 95% LCB reaches Y. A system at Rev% = 100 attains exactly a_h and qualifies without deploying whenever $a_h \geq Y$.

B.3 Process-Aware Success Definition

The main analysis scores an autonomous execution as successful when the four-way verdict matches the reference adjudication (Equation 41). Here we repeat the principal comparisons using the process-aware variant $u_{i,\text{df}}$ (Equation 42), which additionally requires the absence of a disqualifying process defect, to test sensitivity of the routing-quality and frontier conclusions to the deployment-success definition. CliniCARE records the defect as a single per-run indicator, and it is common: 21.9% of the 12,000 runs carry one, ranging from 12.7% (GPT-5.5) to 30.0% (Kimi-K2.7-Code). Because a defect can co-occur with a correct verdict, substituting $u_{i,\text{df}}$ for u_i lowers autonomous success by 4.5 to 13.1 percentage points depending on the system.

The conclusions of Section 5.3 survive the substitution; the frontier of Section 5.4 does not. Routing quality is measuring the same underlying property under either predicate: the Spearman correlation between AUROC under u and under u_{df} across the 16 systems is 0.87, and per-system AUROC moves by at most 0.032 (Table 6). Qwen-3.7-Plus remains among the best-ranking signals and Gemini-3.1-Pro the worst. The deployment consequences, by contrast, shift substantially. Mean review burden across the sweep rises from 53.0% to 69.8%, and at $Y = 0.76$ both Gemini Flash systems move from roughly 38% review to 100%: under a process-aware success definition they have no reachable operating point that deploys at all. This strengthens rather than weakens the comparison of Section 5.6, since the pair contrasted there collapses to two undeployable systems once process defects count against success.

Two further observations. The count of qualifying policies *rises* under u_{df} (488 of 496 against 469), for the same selection reason discussed in Appendix B.2: a harder success predicate forces the development-set optimization to far more conservative thresholds, and those policies clear the held-out bound more comfortably. GLM-5.2 is the visible case, moving from not qualifying at 10.9% review under u to qualifying at 44.8% review under u_{df}. A system's qualified status is therefore not a monotone function of how demanding the success predicate is, and reporting review burden alongside it remains necessary.

	a_0		AUROC		Review%		Qualification under u_{df}		
System	u	u_{df}	u	u_{df}	u	u_{df}	$\widehat{R}$	95% LCB	Qual.
`Claude Code`									
Opus 5	75.7	69.6	0.680	0.691	22.4	37.9	0.842	0.813	✓
Sonnet 5	72.5	65.1	0.681	0.687	29.6	41.6	0.841	0.812	✓
`Codex`									
GPT-5.6-Sol	73.1	67.7	0.640	0.631	32.5	44.5	0.830	0.801	✓
GPT-5.6-Luna	69.9	64.3	0.645	0.631	41.6	56.0	0.829	0.802	✓
GPT-5.5	75.7	70.9	0.652	0.664	21.3	35.7	0.852	0.823	✓
GPT-5.4	72.8	68.3	0.601	0.630	39.2	46.9	0.825	0.796	✓
GPT-5.4-mini	67.5	57.3	0.628	0.624	49.1	67.2	0.855	0.832	✓
`Gemini CLI`									
Gemini-3.6-Flash	71.7	58.7	0.619	0.612	38.4	100.0	0.900	0.900	✓
Gemini-3.5-Flash	70.4	58.7	0.618	0.597	38.7	100.0	0.900	0.900	✓
Gemini-3.1-Pro	70.9	58.7	0.563	0.571	100.0	100.0	0.900	0.900	✓
`opencode`									
DeepSeek-V4-Pro	68.5	58.1	0.664	0.652	44.0	68.5	0.865	0.842	✓
DeepSeek-V4-Flash	68.8	56.3	0.602	0.633	37.1	84.5	0.891	0.876	✓
GLM-5.2	75.7	66.1	0.652	0.678	10.9	44.8	0.811	0.781	✓
Qwen-3.7-Plus	66.4	54.1	0.711	0.679	61.9	61.9	0.815	0.789	✓
MiniMax-M3	66.4	55.2	0.676	0.653	33.9	79.7	0.880	0.862	✓
Kimi-K2.7-Code	65.9	53.1	0.661	0.668	67.7	67.7	0.828	0.803	✓

Table 6. Routing quality and held-out qualification under the process-aware success definition. a_0 and AUROC are on S_{dev} ($n = 375$); Review%, $\widehat{R}$ and the LCB are on S_{qual} at $Y = 0.76$, $a_h = 0.90$, with thresholds reselected at $Y + 0.05$ under each predicate. AUROC is computed from the threshold-based risk–coverage curve, so values are directly comparable across the two columns.

B.4 Complete Per-System Results

This section reports the complete versions of the main-text summaries: per-system risk–coverage and qualification results for all evaluated systems, and routing-score distributions with autonomous success broken down by predicted and reference verdict class.

Qualification across targets. Table 7 extends the single-target view of Table 3 to three targets spanning the range in which most systems have reachable deploying operating points. The step-function behaviour noted in Section 5.5 is visible throughout: review burden is flat across adjacent targets for systems whose confidence takes few distinct values, then jumps. Opus 5 moves 0.8 $\rightarrow$ 22.4 $\rightarrow$ 37.9 percent review across the three targets while Gemini-3.1-Pro is pinned at 100% for the upper two.

Routing behaviour by verdict class. Table 8 breaks the qualification partition down by predicted verdict. Pooled across systems, autonomous success is strongly class-dependent — 0.856 for *Yes*, 0.657 for *No*, 0.544 for *Indeterminate: Lack of Data* and 0.336 for *Indeterminate: Medically Ambiguous* — while mean stated confidence varies far less (0.894, 0.891, 0.839, 0.797). Systems are therefore somewhat but not proportionately less confident on the verdicts they get wrong more often.

The result that matters for the policy design of Section 5.2 is that the routing signal retains ranking power *within* every predicted class: pooled within-class AUROC is 0.639 for *Yes*, 0.611 for *No*, 0.614 for *ILD* and 0.668 for *IMA*. The signal is thus not a proxy for the predicted class. A class-agnostic threshold rule is not discarding information that a class-conditional rule would recover, which is the empirical basis for keeping class semantics inside ϕ_w and out of the escalation policy.

B.5 Cost-of-Conflation Control: Always-Review-Indeterminate

The main-text oversight policy acts only on the routing signal and is agnostic to the four adjudication classes (Section 5.2). As a negative control, we contrast it with a policy that instead escalates on the *predicted class*: every indeterminate adjudication is routed to human review regardless of confidence,

$$\pi_{\mathrm{IR}}(s_i, \widehat{y}_i) = \begin{cases} \text{review}, & \widehat{y}_i \in \{\text{Indeterminate: Lack of Data}, \text{Indeterminate: Medically Ambiguous}\}, \\ \text{accept}, & \widehat{y}_i \in \{\text{Yes}, \text{No}\} \ \wedge \ s_i \geq \tau, \\ \text{review}, & \text{otherwise.} \end{cases} \tag{59}$$

This baseline operationalizes the conflation READY is designed to avoid: it treats a task-level abstention, a possibly-correct indeterminate verdict, an *outcome* scored by ϕ_w, as if it were a deployment-level escalation, a *decision* that should depend on the routing signal. We select and freeze τ for π_{IR} exactly as for the signal-only policy, and report the comparison in Table 9.

The control is not more expensive on this benchmark. We had expected the conflation to show up as wasted oversight, and it does not. Averaged over the 496 policies in the sweep, π_{IR} routes 48.9% of cases to review against 53.0% for the signal-only policy, and it requires strictly more review in only 206 of 496 cells. The single-target comparison could be dismissed as threshold-grid overshoot, so the right-hand columns of Table 9 instead report, for each policy, the minimum review burden at which it *achieves* a

System	$Y = 0.71$				$Y = 0.76$				$Y = 0.80$			
	Rev%	$\widehat{R}$	LCB	Q	Rev%	$\widehat{R}$	LCB	Q	Rev%	$\widehat{R}$	LCB	Q
`Claude Code`												
Opus 5	0.8	0.759	0.720	✓	22.4	0.839	0.807	✓	37.9	0.871	0.844	✓
Sonnet 5	12.0	0.817	0.783	✓	29.6	0.856	0.826	✓	41.6	0.881	0.856	✓
`Codex`												
GPT-5.6-Sol	22.1	0.802	0.768	✓	32.5	0.842	0.812	✓	44.5	0.859	0.832	✓
GPT-5.6-Luna	28.3	0.748	0.712	✓	41.6	0.796	0.764	✓	56.0	0.853	0.828	✓
GPT-5.5	1.9	0.774	0.736	✓	21.3	0.843	0.811	✓	38.1	0.877	0.850	✓
GPT-5.4	12.5	0.731	0.693	–	39.2	0.827	0.797	✓	59.2	0.861	0.836	✓
GPT-5.4-mini	29.1	0.800	0.767	✓	49.1	0.850	0.822	✓	67.2	0.879	0.858	✓
`Gemini CLI`												
Gemini-3.6-Flash	38.4	0.850	0.821	✓	38.4	0.850	0.821	✓	100.0	0.900	0.900	✓
Gemini-3.5-Flash	38.7	0.847	0.818	✓	38.7	0.847	0.818	✓	100.0	0.900	0.900	✓
Gemini-3.1-Pro	33.6	0.838	0.808	✓	100.0	0.900	0.900	✓	100.0	0.900	0.900	✓
`opencode`												
DeepSeek-V4-Pro	44.0	0.839	0.810	✓	44.0	0.839	0.810	✓	68.5	0.883	0.863	✓
DeepSeek-V4-Flash	37.1	0.832	0.802	✓	37.1	0.832	0.802	✓	84.5	0.891	0.876	✓
GLM-5.2	1.1	0.716	0.676	–	10.9	0.762	0.725	–	37.6	0.845	0.816	✓
Qwen-3.7-Plus	22.7	0.775	0.739	✓	61.9	0.874	0.851	✓	61.9	0.874	0.851	✓
MiniMax-M3	33.9	0.809	0.777	✓	33.9	0.809	0.777	✓	61.1	0.872	0.849	✓
Kimi-K2.7-Code	29.1	0.806	0.773	✓	67.7	0.860	0.837	✓	67.7	0.860	0.837	✓

Table 7. Held-out qualification across three reliability targets. $N_{\text{qual}} = 375$, $a_h = 0.90$, thresholds selected on S_{dev} at $Y + 0.05$ and frozen. Rev% is the fraction of qualification cases routed to human review, LCB the one-sided 95% bound of Appendix B.1, and Q whether the policy qualifies. The $Y = 0.76$ block reproduces Table 3; the break-even assumption $a_h^{\min}$ is omitted here for width and is reported at $Y = 0.76$ in that table. Every policy that fails to qualify does so under no $a_h \in [0, 1]$.

given reliability on the qualification partition, which removes the grid effect. The control is still not dominated: it needs more review on 10 of 16 systems at $\widehat{R} = 0.80$ and 6 of 15 at $\widehat{R} = 0.85$, with mean differences of -2.2 and -5.2 percentage points.

The reason is visible in Table 8. Indeterminate adjudications really are much less often correct than definite ones on this benchmark (0.493 against 0.757 pooled on the qualification partition), so predicted class carries genuine information about autonomous failure. Escalating on it is not waste, and we report that rather than presenting a control the data does not support.

What the control does cost. The objection to π_{IR} is structural rather than budgetary. Because the class constraint is fixed, the policy exposes one hard floor on review burden per system — no threshold can accept an indeterminate case — so it does not trace a frontier over the reliability range the way Equation 44 does. It also inherits the benchmark's class-success profile as an assumption: it is competitive here only because indeterminate verdicts happen to be unreliable in this cohort, a property of these labels rather than of the deployment, and one that no part of the policy measures or would detect if it changed. The signal-based policy makes the same trade explicit and tunable, and Appendix B.4 shows the routing signal remains informative *within* each class, so the two are complementary sources of information rather than substitutes. A deployment free to use both would do better than either; the claim defended here

System	*Yes*		*No*		*ILD*		*IMA*		
	%	Succ.	%	Succ.	%	Succ.	%	Succ.	$\lvert\{s_i\}\rvert$
`Claude Code`									
Opus 5	42.4	86.8	38.4	71.5	16.0	58.3	3.2	58.3	25
Sonnet 5	43.7	85.4	36.5	73.0	16.0	63.3	3.7	28.6	26
`Codex`									
GPT-5.6-Sol	38.4	91.7	39.2	67.3	18.7	54.3	3.7	42.9	19
GPT-5.6-Luna	34.1	86.7	41.1	61.0	18.4	40.6	6.4	25.0	21
GPT-5.5	41.1	90.9	42.1	68.4	13.9	61.5	2.9	63.6	25
GPT-5.4	35.7	88.8	42.1	65.8	17.9	47.8	4.3	31.2	32
GPT-5.4-mini	39.5	83.8	42.7	60.0	12.3	45.7	5.6	19.0	33
`Gemini CLI`									
Gemini-3.6-Flash	42.9	86.3	39.7	67.1	13.3	66.0	4.0	40.0	5
Gemini-3.5-Flash	44.3	84.9	40.5	66.4	11.7	63.6	3.5	38.5	5
Gemini-3.1-Pro	39.7	87.2	37.1	71.2	21.3	56.2	1.9	71.4	4
`opencode`									
DeepSeek-V4-Pro	38.7	86.2	41.6	62.2	14.7	54.5	4.8	33.3	15
DeepSeek-V4-Flash	49.1	81.0	35.7	70.1	11.7	56.8	3.5	30.8	10
GLM-5.2	45.9	80.8	39.2	68.0	11.2	57.1	3.7	35.7	23
Qwen-3.7-Plus	35.2	87.9	42.4	58.5	14.9	39.3	7.5	35.7	18
MiniMax-M3	40.0	79.3	41.9	62.4	9.9	56.8	8.3	22.6	19
Kimi-K2.7-Code	39.7	84.6	43.5	61.3	9.1	55.9	7.7	24.1	14

Table 8. Predicted-class mix and within-class autonomous success on the qualification partition. % is the share of the 375 cases receiving each predicted verdict; Succ. is four-way verdict accuracy within that share. *ILD* and *IMA* abbreviate the two indeterminate classes of Equation 40. $\lvert\{s_i\}\rvert$ is the number of distinct stated-confidence values the system emits on this partition, the granularity limit discussed in Section 5.3.

is only that the escalation decision should be driven by a measured routing signal rather than inferred from an outcome class.

	At $Y = 0.76$				Min. Review% to achieve $\widehat{R}$			
	Review%				$\widehat{R} = 0.80$		$\widehat{R} = 0.85$	
System	π_τ	π_{IR}	$\widehat{R}_{\text{IR}}$	Qual.	π_τ	π_{IR}	π_τ	π_{IR}
`Claude Code`								
Opus 5	22.4	19.2	0.815	✓	10.4	19.2	37.9	34.4
Sonnet 5	29.6	21.1	0.827	✓	12.0	19.7	29.6	34.7
`Codex`								
GPT-5.6-Sol	32.5	26.1	0.827	✓	22.1	22.4	44.5	41.1
GPT-5.6-Luna	41.6	28.5	0.790	–	56.0	33.6	56.0	54.9
GPT-5.5	21.3	17.6	0.814	✓	15.5	16.8	35.7	28.0
GPT-5.4	39.2	31.2	0.819	✓	29.3	23.2	59.2	44.3
GPT-5.4-mini	49.1	45.9	0.853	✓	29.1	31.2	53.6	45.9
`Gemini CLI`								
Gemini-3.6-Flash	38.4	45.6	0.864	✓	38.4	45.6	43.5	45.6
Gemini-3.5-Flash	38.7	45.3	0.859	✓	38.7	45.3	43.2	45.3
Gemini-3.1-Pro	100.0	52.8	0.897	✓	33.6	23.2	—	52.8
`opencode`								
DeepSeek-V4-Pro	44.0	54.9	0.860	✓	44.0	54.9	68.5	54.9
DeepSeek-V4-Flash	37.1	45.3	0.856	✓	37.1	26.1	84.5	45.3
GLM-5.2	10.9	17.1	0.775	–	32.0	23.5	44.8	40.3
Qwen-3.7-Plus	61.9	38.9	0.820	✓	61.9	30.4	61.9	64.0
MiniMax-M3	33.9	39.2	0.825	✓	33.9	39.2	60.8	64.8
Kimi-K2.7-Code	67.7	37.9	0.818	✓	29.1	33.6	67.7	70.4

Table 9. Signal-based routing against the always-review-indeterminate control, $a_h = 0.90$, $N_{\text{qual}} = 375$. Left block: both policies selected on S_{dev} at $Y + 0.05$ and frozen, evaluated at $Y = 0.76$. Right block: the minimum review burden at which each policy attains the stated reliability among its own reachable operating points on S_{qual}, which removes the threshold-grid overshoot that confounds the single-target comparison; — marks a reliability no operating point of that policy reaches.