Strategic Decision Support for AI Agents

Shayan Kiyani*
University of Pennsylvania
shayank@engineering.upenn.edu

Sima Noorani*
University of Pennsylvania
nooranis@engineering.upenn.edu

George Pappas
University of Pennsylvania
pappasg@engineering.upenn.edu

Hamed Hassani
University of Pennsylvania
hassani@engineering.upenn.edu

Abstract

Traditionally, decision support studies how humans use machine learning models to make better decisions. In modern agentic systems, this division of roles is increasingly reversed: AI agents act on behalf of users, while humans and tools becomes support mechanisms around them. This role reversal brings reliability concerns to the forefront, since agentic errors can be consequential and agent behavior must remain aligned with human goals and constraints. Departing from the classical view of decision support, we revisit its two basic principles, the cost–value tradeoff of seeking support and the role of uncertainty quantification, in a setting where AI agents are the central actors. We propose a framework for *strategic decision support* for AI agents through an optimization problem that minimizes support usage subject to controlling a counterfactual missed-support error: the probability that the agent acts alone on instances where support would have materially improved its output. At the population level, we show that the optimal policy is a threshold rule on the *value of support*. Building on this structure, we develop an online algorithm that adaptively thresholds such a score and uses randomized exploration to control missed-support error without distributional assumptions. We further introduce a calibration-on-the-fly method that reduces unnecessary support calls online. We instantiate this framework across diverse scenarios, including information gathering, human–AI collaboration, and tool use, showing how each can be modeled through the same strategic decision-support lens. Experiments across these settings show that our method reliably controls the target error while substantially reducing support usage in practice.

1 Introduction

Decision support has long been studied in settings where a human decision maker is aided by machine learning models that provide predictive guidance. In such systems, the model serves as a support tool, while the human remains the final actor. Today, however, modern AI systems, such as large language models, are increasingly deployed in a different role: as autonomous agents that must act on behalf of users in complex and uncertain environments. As a result, humans, domain-specific tools, and auxiliary sources of information are themselves becoming support mechanisms around AI agents.

This shift in roles brings reliability concerns to the forefront. As AI agents take actions, their errors become more consequential: they may execute code that overwrites critical records, trigger financial transactions that move funds incorrectly, or recommend medical actions that expose patients to harm. At the same time, as users offload more decisions to AI systems, it becomes critical to ensure that

*Equal contribution. Correspondence to: shayank@engineering.upenn.edu, nooranis@engineering.upenn.edu.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the agent's behavior remains aligned with human goals and constraints. These challenges call for rethinking a basic question of decision support::

When should an AI agent act alone, and when must it seek support to avoid consequential errors or misalignment with downstream intent?

Addressing this question requires reexamining the two principles underlying decision support systems.

Cost–value tradeoff. In the classical view, the support system, often an ML model, is relatively cheap to query but imperfect, since its predictions are prone to mistakes. In the agentic setting, the reversal of roles creates a different equilibrium: *support is often costly to seek* (e.g., requiring human effort or additional compute and latency), but can be *reliable and instrumental* for avoiding decision errors and keeping the agent aligned with human intent.

Uncertainty quantification. At the core of any AI agent taking action lies uncertainty quantification. For decision support, the right lens is to quantify *whether costly support is likely to substantially improve the agent's output.* Thus, uncertainty is a joint property of the agent *and* the support mechanism. This differs from the common view, which frames uncertainty through the agent alone: whether the agent is confident in its action, or whether the environment and task are well-understood.

To this end, we propose a formulation that separates the *value of support* from its cost and treats support-seeking as a strategic decision. The value of support captures whether the supported output is better than the output the agent would have produced alone. The central error is then *missed support*, which occurs when the agent acts alone even though support would have improved its output. This error is counterfactual, since we only learn whether support would have helped when support is actually called. Strategic decision support therefore becomes an optimization problem: use support as rarely as possible, while controlling the missed support error. We summarize our contributions below. Figure 1 illustrates a missed-support error (top) and its correction via strategic decision support oversight (bottom).

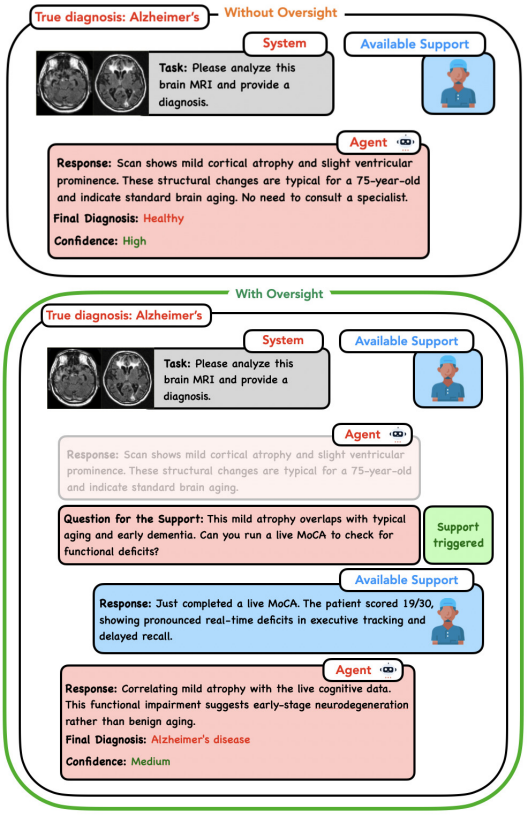

Figure 1: Effect of strategic decision support oversight. Top: Without oversight, the agent confidently misdiagnoses an Alzheimer's case as healthy and fails to seek available support, resulting in a missed-support error. Bottom: With oversight, support is triggered and the agent consults a clinician to gather additional interactive tests, enabling the correct diagnosis to be recovered.

1. A framework for strategic decision support. In Section 3, we introduce SDS-Opt, an optimization problem that minimizes the rate of support calls subject to controlling the missed-support error. This formulation is built around the *value of support*: whether the supported output would materially improve over the output the agent would have produced alone. At the population level, we show that the optimal strategy has a simple threshold form over value of support.

2. An online algorithm with distribution-free error control. In Section 4, we build on this structure to develop an online algorithm that adaptively thresholds such a score and uses randomized exploration to control the counterfactual missed-support error without distributional assumptions. We further introduce a calibration-on-the-fly method that improves the score over time, reducing unnecessary support calls while

preserving the validity of the thresholding guarantee.

3. Applications and empirical validation. Our framework also provides a common language for designing support around AI agents. We instantiate this perspective through four representative categories that serve as running examples throughout the paper:

- **Ex1: Information gathering.** The agent must act from incomplete information, while additional evidence could improve its output at a cost. Support may consist of follow-up questions, additional evidence, or expert-provided details. For example, a medical assistant may reason from an initial symptom description, but further history, examination findings, or laboratory results may materially change its recommendation.
- **Ex2: Human-in-the-loop planning.** The agent has general task knowledge, but lacks local context needed to produce an appropriate plan. Support may consist of user preferences, environment-specific constraints, or object locations. For example, a household robot may know how to clean a room in general, but may need the resident's input about which items are fragile, which areas to avoid, or what should be moved before cleaning.
- **Ex3: Human-AI collaborative reasoning.** The agent can attempt the problem on its own, but may become uncertain about a specific reasoning step. Support may consist of checking an approach, identifying an error, or suggesting a relevant technique. For example, a math-solving agent may ask a human to verify a key computation or point out where the argument breaks down when its reasoning appears unreliable.
- **Ex4: Tool use.** The agent faces a task where external tools provide reliable information or computation that it may not perform correctly on its own. Support may consist of executing code, querying a database, or searching the web. For example, an agent answering a question about a table may call a SQL engine rather than relying only on its internal reasoning.

In Section 5, we instantiate these categories on real-world datasets and LLM agents[2], showing that our method controls the missed-support error while substantially reducing unnecessary support calls across all four categories; Figure 2 previews our results on one model per application, with the full empirical study deferred to Section 5.

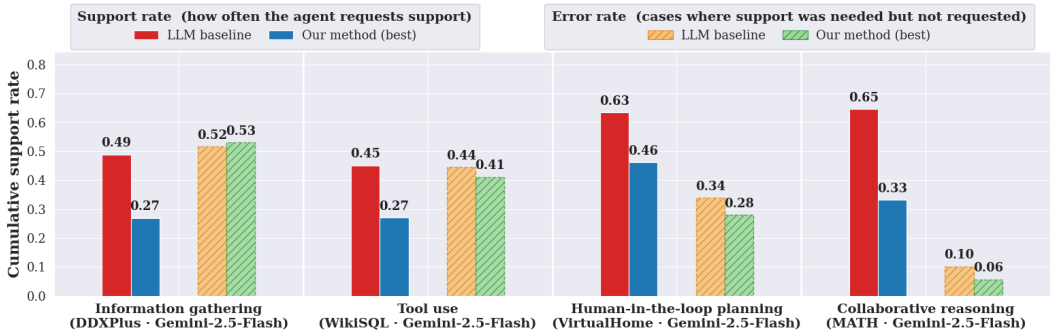

Figure 2: **Our method invokes decision support substantially less often than an LLM-decides baseline, while matching its error rate.** For each of four agentic applications: information gathering (DDXPlus), tool use (WikiSQL), human-in-the-loop planning (VirtualHome), and collaborative human–AI reasoning (MATH), all using Gemini-2.5-Flash, we report two pairs of bars. The left pair (solid) shows the cumulative *support rate*: the fraction of inputs on which the agent requests external support. The right pair (hatched) shows the cumulative *missed support error rate*: cases where support would have materially improved the output but was not requested. In each pair, the left bar is the LLM baseline (the agent deciding for itself when to ask for support) and the right bar is our best learned method. Across all four applications, our method invokes support far less often than the baseline while maintaining a comparable or lower error rate.

[2]The code is publicly available at `https://github.com/nooranisima/strategic-decision-support`.

2 Related Works

We briefly discuss closely related works here and defer a more detailed dicussion to Appendix A.

Training agents to seek support. One body of work bakes support-seeking behavior into the agent itself through training, both on the tool-use and retrieval side [5, 28, 32, 33, 61, 70] and on the user-interaction side [2, 42, 85, 92]. These methods produce stronger agents: a model already knows when to call SQL, or that has been fine-tuned to ask the right clarifying questions, is precisely the kind of base policy our framework sits on top of. Our experiments use frontier models that already incorporate substantial training of this kind, and our gains reflect what is achievable as an oversight layer on top of agents that are already strong at the underlying support modalities.

Inference-time policies for support-seeking. A closely related line of work designs inference-time policies that decide whether an agent should pause to seek information before acting. Some threshold the agent's own confidence in its answer [39, 68, 85]; others compute the expected utility of asking a clarification under an explicit cost [19]; and others use offline-calibrated prediction sets over candidate user intents to trigger help-seeking [67]. We instead provide a unifying framework that brings these approaches under the same design principles, through an oversight layer with rigorous finite-sample error control. This enables us to handle a broad set of support modalities within a single algorithm, operate fully online, and control the counterfactual missed-support error at a user-chosen level which we test in our experiments across information gathering, tool use, and human-AI collaboration.

3 Fundamentals of Strategic Decision Support

In this section, we introduce the central objectives of strategic decision support at the population level. We then characterize the optimal support-seeking strategy in this regime. These results form the foundation for the practical sequential algorithm developed in Section 4, where the population quantities are unknown and support decisions must be made online.

We begin by modeling the interaction between an AI agent and a decision-support mechanism. Let $X \sim P_X$ denote all information available to the agent at decision time, including the user prompt, task description, and any other available input modalities. Given $X = x$, the agent can first produce an unsupported output $Y_0 \sim \pi(\cdot \mid x, \texttt{"no support"})$, corresponding to the response it would generate on its own. After observing this initial output, the agent must decide whether to settle for it or seek support.[3] A support-seeking strategy is therefore a function $a : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, where $a(x, y_0) = 0$ means that the agent acts on its own and keeps the unsupported output, while $a(x, y_0) = 1$ means that it seeks support and produces a supported output $Y_1 \sim \pi(\cdot \mid x, y_0, \texttt{"with support"})$.

For example, in Ex1, x may contain a patient's initial symptoms and description of their condition, while y_0 is the medical assistant's initial recommendation. The strategy then decides whether to proceed with this recommendation or seek additional clinical evidence before producing a supported response. This abstraction lets us formalize both the value and the cost of support.

Value of support. We begin by introducing a *value indicator*

$$g(X, Y_0, Y_1) \in \{0, 1\},^{4}$$

which indicates whether, after comparing the unsupported outcome Y_0 and the supported outcome Y_1, support is judged to have been materially beneficial on that instance. Thus, $g = 1$ means that support helped, while $g = 0$ means that it did not. This notion is deliberately separate from the cost of seeking support: g evaluates only whether the supported outcome is better, according to the provider's or downstream user's notion of performance.

This indicator induces the central population quantity in our framework, the *value of support*,

$$\mathrm{val}(x, y_0) := \mathbb{P}(g(X, Y_0, Y_1) = 1 \mid X = x,\, Y_0 = y_0)\,.$$

In words, $\mathrm{val}(x, y_0)$ is the probability that calling support would produce a substantially better outcome after observing the input instance and the agent's unsupported output.

[3] One may avoid or only partially generate the unsupported output (Y_0) before deciding whether to seek support. Our framework and guarantees are robust to such choices; we discuss these further in Section 4 and Appendix C.12.

[4] One may similarly define a continuous version of g indicating how much value is added, and define the value of support as its expectation rather than the probability of being 1.

The definition of g depends on what it means for support to materially improve the output. In Ex1, Ex3, and Ex4, where the output has a verifiable final answer, a natural choice is to set $g = 1$ only when Y_0 is incorrect and Y_1 is correct, and to set $g = 0$ in all other cases. In EX2, where the output is a structured plan, g can instead indicate whether the supported output improves a task-specific quality metric beyond a chosen threshold. More generally, when the support mechanism is reliable, the simpler choice of $g = 1$ whenever $Y_1 \neq Y_0$ is often a strong proxy, since a change in output is itself indicative of a material difference. We discuss precise implementations of g in Section 5.2.

Error of missed support. From the viewpoint of providers and downstream decision-makers, missing support when it would have helped is often the most consequential failure, since better outputs may directly translate into better downstream decisions. Thus our guiding principle is that support should be used whenever it would materially improve the output. This leads to an asymmetric notion of error: what matters is how often the agent fails to seek support on the very instances where support would have helped. For a strategy a, we therefore define the *missed-support error* as

$$\mathbb{P}\big(a(X, Y_0) = 0 \mid g(X, Y_0, Y_1) = 1\big).$$

It measures the probability that the agent acts alone, conditioned on support being beneficial.

Cost of support. Support, however, is not free. It may require additional latency, computation, API usage, or human effort. Thus, the trivial strategy that always seeks support is typically unacceptable, even though it drives the missed-support error to zero. The goal is therefore to control missed-support error while using support only when necessary. To capture this, we measure the cost of a strategy a by its support rate

$$\mathbb{E}\big[\mathbf{1}\{a(X, Y_0) = 1\}\big],$$

namely, the population probability that the agent seeks support. [5]

Putting these pieces together, we arrive at the following population-level formulation. We seek a strategy that minimizes how often support is used, while guaranteeing that the agent rarely skips support on the instances where support would have been beneficial:

Strategic Decision Support Optimization

$$\begin{aligned} \underset{a:\, \mathcal{X}\times\mathcal{Y}\to\{0,1\}}{\text{minimize}} \quad & \mathbb{E}_X\big[\mathbf{1}\{a(X, Y_0) = 1\}\big] \\ \text{subject to} \quad & \mathbb{P}\big(a(X, Y_0) = 0 \mid g(X, Y_0, Y_1) = 1\big) \;\leq\; \varepsilon \end{aligned} \tag{SDS-Opt}$$

The parameter $\varepsilon \in [0, 1]$ specifies the tolerated level of missed-support error. Smaller values of ε require the strategy to seek support on a larger fraction of instances where support would help, and therefore lead to more frequent support calls. Hence, this formulation captures the balance: support should be used sparingly, but not at the expense of missing the instances where it is genuinely needed.

We now characterize the optimal solution of (SDS-Opt).

Theorem 3.1. *There exists an optimal solution to (SDS-Opt) of the form*

$$a^\star(x, y_0) = \mathbf{1}\{\mathrm{val}(x, y_0) > \tau^\star\},$$

with possible randomization on the boundary , i.e. when $\{\mathrm{val}(x, y_0) = \tau^\star\}$*, if needed.*

Theorem 3.1 identifies $\mathrm{val}(x, y_0)$ as the fundamental quantity for support seeking. The optimal strategy thresholds this quantity, i.e., support should be sought when its value is "high enough."

This characterization forms the foundation of our online method. Since $\mathrm{val}(x, y_0)$ is generally unknown in practice, the core task becomes to approximate it and threshold it adaptively. The algorithm in the next section does exactly this: it uses a score function as a proxy for the value of support, refines this score through calibration-on-the-fly, and updates a decision threshold online to determine when support should be sought.

[5] We define cost as the frequency of support calls. In practice, however, the cost of support may depend on the instance. We defer these finer-grained formulations to future work.

4 Online Algorithm and Guarantees

In this section, we design *Strategic Oversight for Support-seeking* (SOS), an online algorithm for deciding when an AI agent should seek support. The population result of Section 3 suggests a simple principle: estimate the value of support and seek support when this value is high enough. SOS turns this principle into a rigorous online procedure, where the value of support is estimated and calibrated online, and support decisions are made based on a threshold that is debiased sequentially.

We fix an AI agent and a support mechanism, and make no distributional assumptions on the data, the agent's behavior, or the behavior of the support mechanism. Building on the structural result of Section 3, SOS uses a score function

$$s_\theta : \mathcal{X} \times \mathcal{Y} \to [0, 1],$$

where $s_\theta(x, y)$ evaluates an input x together with a candidate output y. The score is intended to approximate the value of support, $\mathrm{val}(x, y)$, from Theorem 3.1. Here, θ denotes the parameters of the score function; these parameters may be fixed in advance, pretrained, or learned online as feedback is collected. Concrete choices and parameterizations of the score function are discussed in Section 5.

Online interaction pipeline. At each round $t = 1, 2, \ldots$:

1. The agent receives an input x_t, produces an unsupported output y_0^t, and computes a score
$$s_t := s_{\theta_t}(x_t, y_0^t).$$
2. Based on s_t, the policy chooses an action $a_t \in \{0, 1\}$, where $a_t = 0$ means proceeding without seeking support, and $a_t = 1$ means seeking support.
3. If $a_t = 0$, the agent finalizes the unsupported outcome y_0^t and proceeds to the next round.
4. If $a_t = 1$, the agent seeks support, produces the supported outcome y_1^t, and computes
$$g_t := g(x_t, y_0^t, y_1^t) \in \{0, 1\},$$
which indicates whether support was beneficial on round t.

Our goal is to control the empirical missed-support error at a user-specified level $\alpha \in (0, 1)$: [6]

$$\widehat{\mathrm{MSE}}(T) := \frac{\sum_{t=1}^{T} g_t(1 - a_t)}{\sum_{t=1}^{T} g_t} \leq \alpha. \tag{1}$$

This error is the fraction of beneficial-support rounds on which the policy nevertheless acts alone. Two remarks are in order.

- We modeled the online interaction in a form where the support decision is made using both the input x_t and the unsupported output y_0^t. In practice, one may choose to avoid generating, or only partially generate, the unsupported outcome before deciding whether to seek support. On rounds where support is sought, one may likewise be able to compute the value of g without generating all of y_0^t. Our framework is easily tuned to such operational choices. We discuss these practical variants and their implications for cost and performance in more detail in Appendix C.12.
- The error we seek to control depends on g_t, but g_t is only revealed on rounds where support is sought, since computing it requires comparing y_0^t and y_1^t. Thus, the relevant error is fundamentally counterfactual and only selectively observed. This partial-feedback structure is exactly what makes randomization necessary in the online algorithm.

Algorithm. At each round t, we maintain a threshold λ_t and define the support-seeking probability

$$p_t := \begin{cases} 1, & s_t \geq \lambda_t, \\ \mu, & s_t < \lambda_t, \end{cases}$$

where $\mu \in (0, 1)$ is a fixed exploration parameter. The action is then sampled as $a_t \sim \mathrm{Bernoulli}(p_t)$.

[6] We adopt the convention $\widehat{\mathrm{MSE}}(T) = 0$ when $\sum_{t=1}^{T} g_t = 0$.

Algorithm 1 Strategic Oversight for Support-seeking (SOS)

Require: Target $\alpha \in (0,1)$; exploration parameter $\mu \in (0,1)$; step sizes $\{\eta_t\}_{t\geq 1}, \{\gamma_t\}_{t\geq 1}$; initial threshold λ_1; initial score parameter θ_1

for $t = 1, 2, \ldots$ **do**

Receive input x_t and generate y_0^t

Compute score $s_t = s_{\theta_t}(x_t, y_0^t) \in [0,1]$

Support-seeking probability and action

$p_t \leftarrow \mu + (1-\mu)\,\mathbf{1}\{s_t \geq \lambda_t\}$

Sample $a_t \sim \mathrm{Bernoulli}(p_t)$

Decision and feedback

if $a_t = 0$ **then**

$\lambda_{t+1} \leftarrow \lambda_t, \theta_{t+1} \leftarrow \theta_t$

else

Seek support and compute y_1^t

Compute $g_t = g(x_t, y_0^t, y_1^t) \in \{0,1\}$

Threshold update and calibration-on-the-fly

$$\lambda_{t+1} \leftarrow \lambda_t - \eta_t \frac{g_t}{p_t}\Big((1-p_t)\mathbf{1}\{s_t < \lambda_t\} - \alpha\Big)$$

$$\theta_{t+1} \leftarrow \theta_t - \gamma_t \frac{a_t}{p_t}\nabla_\theta \big(s_\theta(x_t, y_0^t) - g_t\big)^2\Big|_{\theta=\theta_t}$$

end if

end for

Thus, when the score exceeds the current threshold, the algorithm always seeks support; when the score falls below the threshold, it still seeks support with a small probability μ in order to obtain feedback about whether support would have been helpful.

On rounds where support is sought, we observe g_t, and the threshold is updated as

$$\lambda_{t+1} \leftarrow \lambda_t - \eta_t \cdot \frac{g_t a_t}{p_t}\Big[(1-p_t)\mathbf{1}\{s_t < \lambda_t\} - \alpha\Big],$$

where $\eta_t > 0$ is a step size and $\alpha \in (0,1)$ is the target missed-support error level.

This update resembles online quantile-tracking: the quantity inside the brackets plays the role of an error-minus-target signal. In a deterministic threshold policy, this signal would be $\mathbf{1}\{s_t < \lambda_t\} - \alpha$, since rounds below the threshold are exactly those on which the agent proceeds without support. Here, however, the action is randomized: even when $s_t < \lambda_t$, support is still sought with probability $p_t = \mu$. Thus the realized missed-support event is not simply $\mathbf{1}\{s_t < \lambda_t\}$, but rather $(1-a_t)\mathbf{1}\{s_t < \lambda_t\}$. Since the only randomness comes from the algorithm's internal exploration, we have

$$\mathbb{E}[(1-a_t)\mathbf{1}\{s_t < \lambda_t\} \mid \mathcal{F}_{t-1}, x_t, s_t] = (1-p_t)\mathbf{1}\{s_t < \lambda_t\},$$

where the expectation is taken only over the Bernoulli draw $a_t \sim \mathrm{Bernoulli}(p_t)$. This explains the extra factor $(1-p_t)$ in the update. The prefactor $g_t a_t / p_t$ then serves as an importance-weighted correction for the fact that g_t is observed only on rounds where support is sought.

Calibration-on-the-fly: Alongside the threshold update, we also allow the score itself to be updated online. Specifically, we treat the score as parameterized by θ_t and update

$$\theta_{t+1} \leftarrow \theta_t - \gamma_t \frac{a_t}{p_t}\nabla_\theta (s_\theta(x_t) - g_t)^2\Big|_{\theta=\theta_t},$$

where $\gamma_t > 0$ is a step size. The factor a_t/p_t again provides the appropriate importance-weighted correction, since g_t is observed only on rounds where support is sought. At this level, we view this as an abstract online calibration procedure for improving how well the score tracks the latent value of support; concrete score parameterizations and design choices will be discussed in the next section.

The next result shows that the threshold update rule in Algorithm 1 yields a distribution-free finite-sample guarantee for controlling the empirical missed-support error.

Theorem 4.1. *Run Algorithm 1 with constant step size $\eta_t \equiv \eta > 0$, exploration parameter $\mu \in (0,1)$, target level $\alpha \in (0, 1-\mu)$, and initial threshold $\lambda_1 \in [0,1]$. Fix any horizon $T \geq 1$, and define $N_g(T) := \sum_{t=1}^T g_t$. Then for any $\delta \in (0,1)$, w.p. at least $1-\delta$ over the algorithm's randomness,*

$$\widehat{\mathrm{MSE}}(T) \leq \alpha + \Delta(N_g(T), \delta),$$

where, for $N \geq 1$, $\Delta(N,\delta) := \frac{1+2\eta/\mu}{\eta N} + \sqrt{\frac{8\log(4/\delta)}{\mu N}} + \frac{4\log(4/\delta)}{3\mu N}$, and $\Delta(0,\delta) := 0$.

Theorem 4.1 shows that the empirical missed-support error is controlled at the target level α up to a finite-sample term. The bound has two qualitatively different sources. The first term, of order $1/(\eta N)$, is the intrinsic error of online quantile tracking and would remain even if the value of support signal were always revealed[3, 27, 65]. The remaining terms arise from the randomized exploration needed to obtain unbiased feedback. The dependence on μ makes the tradeoff explicit: larger exploration improves error control, but increases support usage. Finally, the validity guarantee comes from the threshold update rule and does not rely on calibration-on-the-fly, so it holds whether the score parameters are fixed or getting updated. The role of calibration-on-the-fly is instead to improve the score over time and reduce unnecessary support calls, as we demonstrate empirically.

5 Experiments

This section is organized as follows. Section 5.1 introduces three families of score designs. Section 5.2 describes the four tasks, datasets, and baseline methods. Section 5.3 empirically validates Theorem 4.1, showing that the algorithm controls the missed-support error at the target level α across all score families. Section 5.4 then fixes the missed-support error across methods and compares how often each invokes support. At matched error, the method calling support less often better identifies when support is needed, and the gap between methods reflects the value of richer score signals.

5.1 Score Design

Algorithm 1 is agnostic to how the score $s_\theta(x)$ is constructed. Since the threshold update controls the missed-support error at level α regardless of the score's quality (Theorem 4.1), the score's role is to reduce how often support is called by separating instances where support would be beneficial from those on which it would not. We evaluate three score families, organized in increasing order of expressivity.

- **Confidence score.** The simplest family takes the score directly from the black-box signal, with no learnable parameters or calibration-on-the-fly: $s(x) := \hat{g}_{\text{bb}}(x)$. The signal is thresholded as-is, and only λ in Algorithm 1 is adapted online. This natural training-free baseline serves as a reference for parameterized scores, and reappears as the anchor in the anchored family below.
- **Representation score.** The second family parameterizes the score as a linear function of a frozen embedding, $s_\theta(x) := \sigma(\theta^\top \phi(x))$. $\phi(x) \in \mathbb{R}^d$ embeds the input, $\sigma(\cdot)$ is the sigmoid that keeps the score in $[0, 1]$, and only the linear coefficients $\theta \in \mathbb{R}^d$ are updated by calibration-on-the-fly. This is a linear probe over a representation that summarizes the input without committing to a particular notion of confidence; calibration-on-the-fly learns which directions in representation space predict whether support helps. The embedding ϕ can be any pretrained text representation. In the *black-box* setting, ϕ comes from a separate frozen encoder applied to the input. In the *white-box* setting, we can also use the LLM's hidden states. The input to ϕ is itself a modeling choice; we use the prompt x throughout and study alternatives (such as also including y_0 when generated) in Appendix C.12.
- **Anchored score.** The final family combines the parameterized linear term with the black-box signal in logit space: $s_\theta(x) := \sigma\big(\text{logit}(\hat{g}_{\text{bb}}(x)) + \theta^\top \phi(x)\big)$. Here $\hat{g}_{\text{bb}}$ acts as an *anchor* providing an initial estimate of the value of support, and $\theta^\top \phi(x)$ learns a residual correction in logit space. When the anchor is already well-aligned with g, the anchored score inherits its quality and calibration-on-the-fly need only learn small adjustments. When the anchor is uninformative or systematically biased, the linear term can override it.

The Representation and Anchored families are each instantiated with three embedding choices, summarized below. Black-box (BB) variants apply a separate frozen encoder to the input; the white-box (WB) variant uses the LLM's own hidden state at the final input token.

Variant	BB	WB	Embedding model
CONFIDENCE	✓	✓	— (no embedding; $s(x) = \hat{g}_{\text{bb}}(x)$)
REP-MINILM, ANC-MINILM	✓		`all-MiniLM-L6-v2` sentence encoder
REP-GEMINI, ANC-GEMINI	✓		`gemini-embedding-2` text encoder
REP-HIDDEN, ANC-HIDDEN		✓	LLM's hidden state at the final input token

5.2 Tasks and Baselines

Tasks. We instantiate each of the application categories from Section 1 on a concrete dataset. Task specifications and prompts are deferred to Appendix C.8.

Information gathering on **DDXPlus** [77](medical diagnosis): A diagnostic agent receives a patient's chief complaint and initial symptoms as x; support reveals follow-up questions, examination findings, and laboratory results. We set $g = 1$ when y_0 is an incorrect diagnosis and y_1 is correct against the ground-truth pathology.

Tool use on **WikiSQL** [93]: An agent receives a natural-language question about a table as x; support consists of formulating and executing SQL queries against it. We set $g = 1$ when y_0 is incorrect and y_1 matches the dataset's expected answer.

Human-in-the-loop planning on **VirtualHome**[59]: A household robot receives a task description as x and must output a plan as a sequence of actions; support reveals scene-specific object locations and constraints from the resident. $g = 1$ when the longest-common-subsequence (LCS) overlap of y_1 with the gold action sequence exceeds that of y_0.

Human-AI collaborative reasoning on Level 4–5 problems from **MATH** [30]: An reasoning agent attempts solving math problems; support comes from a stronger reasoner that provides targeted guidance on a step the agent identifies as uncertain, without revealing the full solution. We set $g = 1$ when y_0 is incorrect and y_1 is correct.

We run all four tasks with three frontier base agents spanning both regimes: Qwen-2.5-7B [63] (white-box), Gemini-2.5-Flash [22] (black-box), and GPT-4o-mini [56] (black-box).

Baselines. Our primary baseline is **LLM-decides**, which lets the LLM itself choose whether to seek support after producing y_0. This is a natural reference point as it isolates what the agent does on its own, with no oversight layer, no error target, and no calibration. Its missed-support rate is a property of the model and the task, and cannot be controlled. We take this rate as the target α for our algorithm on each (task, model) pair, so that all comparisons are on equal footing. We then run each score variant from Section 5.1 inside Algorithm 1: CONFIDENCE adapts only λ_t, while the Representation and Anchored variants additionally update the score parameters θ via calibration-on-the-fly.

5.3 Error Control

We track the cumulative missed-support error $\mathrm{MSE}(T)$ (Equation (1)), the empirical fraction of rounds where support would have helped but was not invoked. Figure 3 shows the running $\mathrm{MSE}(T)$ for one representative task–model pair with the x-axis indicating progress through the interaction stream as a fraction of total rounds. All score variants converge to the target α (the rate achieved by LLM-Decides), with trajectories following the typical adaptation profile of online quantile tracking. Convergence rate is governed by the threshold step size η, which trades off adaptation speed against stability. The same behavior hold across all tasks and base agents. The full set is reported in Appendix C.6.

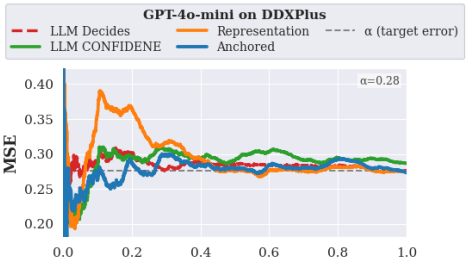

Figure 3: Cumulative missed-support error on DDXPlus with GPT-4o-mini as the agent.

5.4 Support Efficiency

Error control alone does not imply an efficient support policy. A policy that always invokes support has zero missed-support error but is maximally wasteful. We therefore compare methods on equal footing by asking how often each invokes support at the same missed-support error level. Figure 4 reports the cumulative support rate across all task–model pairs, with the target α for SOS set to the missed-support rate achieved by LLM-DECIDES on each pair. Two general patterns emerge.

Calibration-on-the-fly recovers from uninformative signals. The Representation family reliably reduces the support rate relative to LLM-DECIDES on all task–model pairs, and the Anchored family does so on 11 of 12, regardless of how informative the underlying signal is. The oversight layer does

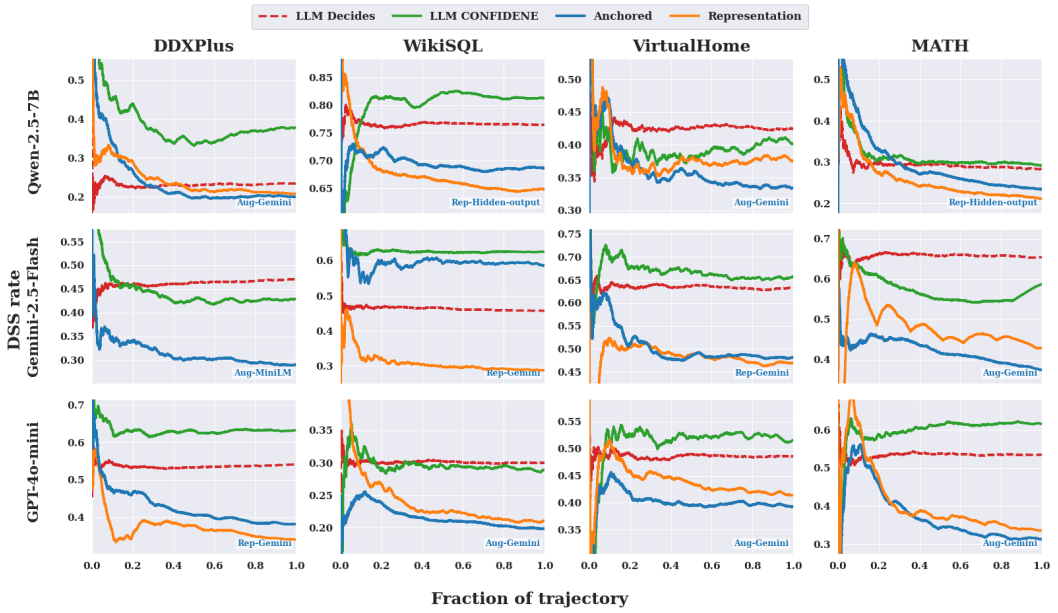

Figure 4: Cumulative support rate $\widehat{\mathrm{SR}}_T = \frac{1}{T}\sum_{t=1}^{T} a_t$ across all task–model pairs at matched missed-support error. We show the best-performing variant across both families, paired with its same-embedding counterpart from the other family. Full per-panel comparisons in Appendix C.1.

not require a strong starting signal, since calibration-on-the-fly improves it from feedback on rounds where support is invoked. The Confidence score is unstable across tasks and agents; sometimes below LLM-DECIDES (e.g., VirtualHome on Qwen-2.5-7B), sometimes well above (e.g., WikiSQL on Qwen-2.5-7B). The parameterized families absorb this variability and reach lower support rates.

Anchoring on a reliable initial signal accelerates the gain. The two parameterized families differ in whether they treat the initial signal as a useful prior or discard it. When the anchor is informative, calibration-on-the-fly only needs to learn a small correction, and the Anchored variants outperform their Representation counterparts (e.g., MATH on Gemini-2.5-Flash). When the anchor is uninformative or actively misleading, the Representation family pulls ahead, since the residual term in the Anchored score has to first undo the bad anchor before it can learn (e.g., WikiSQL on Gemini-2.5-Flash). In our experiments the anchor is the LLM's own self-confidence, but the same principle applies to any auxiliary signal.

Takeaway. AI agents struggle to determine when seeking support is beneficial, even when they are proficient at utilizing that support once provided. Our strategic oversight algorithm significantly outperforms an agent's self-decisions, achieving lower support rates while maintaining formal error control. Among score designs, parameterized scores that learn from representations perform reliably well, with calibration-on-the-fly effectively reducing the support rate as the query stream progresses.

6 Limitations and Future Work

We focus on a binary support-seeking decision and measure cost by the frequency of support calls. In practice, agents may have multiple support options, and the cost of support may vary across options and instances. One may also replace the binary value indicator g with a continuous measure of how much value support adds, defining the value of support as its expectation rather than the probability that $g = 1$. We defer these finer-grained formulations to future work.

Acknowledgments and Disclosure of Funding

The authors thank EnCORE, the Institute for Emerging CORE Methods in Data Science, for their support, as well as NSF award 2502489 under the MFAI: Mathematical Foundations of Alignment in Generative Artificial Intelligence program. SK additionally acknowledges support from a gift from AWS to Penn Engineering's ASSET Center for Trustworthy AI.

References

[1] Lukas Aichberger, Kajetan Schweighofer, Mykyta Ielanskyi, and Sepp Hochreiter. Semantically diverse language generation for uncertainty estimation in language models, 2024. URL https://arxiv.org/abs/2406.04306.

[2] Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. Star-gate: Teaching language models to ask clarifying questions, 2024. URL https://arxiv.org/abs/2403.19154.

[3] Anastasios N. Angelopoulos, Emmanuel J. Candes, and Ryan J. Tibshirani. Conformal pid control for time series prediction, 2023. URL https://arxiv.org/abs/2307.16895.

[4] Adrian Arnaiz-Rodriguez, Nina Corvelo Benz, Suhas Thejaswi, Nuria Oliver, and Manuel Gomez-Rodriguez. Towards human-ai complementarity in matching tasks, 2025. URL https://arxiv.org/abs/2508.13285.

[5] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

[6] Varun Babbar, Umang Bhatt, and Adrian Weller. On the utility of prediction sets in human-ai teams, 2022. URL https://arxiv.org/abs/2205.01411.

[7] Gagan Bansal, Besmira Nushi, Ece Kamar, Eric Horvitz, and Daniel S. Weld. Is the most accurate ai the best teammate? optimizing ai for teamwork, 2021. URL https://arxiv.org/abs/2004.13102.

[8] Nina L. Corvelo Benz and Manuel Gomez Rodriguez. Human-alignment influences the utility of ai-assisted decision making, 2025. URL https://arxiv.org/abs/2501.14035.

[9] Sebastian Bordt and Ulrike Von Luxburg. A bandit model for human-machine decision making with private information and opacity. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 7300–7319. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/bordt22a.html.

[10] Lawrence Chan, Dylan Hadfield-Menell, Siddhartha Srinivasa, and Anca Dragan. The assistive multi-armed bandit, 2019. URL https://arxiv.org/abs/1901.08654.

[11] Mohammad-Amin Charusaie, Hussein Mozannar, David Sontag, and Samira Samadi. Sample efficient learning of predictors that complement humans, 2022. URL https://arxiv.org/abs/2207.09584.

[12] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023. URL https://arxiv.org/abs/2305.05176.

[13] John J. Cherian, Isaac Gibbs, and Emmanuel J. Candès. Large language model validity via enhanced conformal prediction methods, 2024. URL https://arxiv.org/abs/2406.09714.

[14] Bo Cowgill and Megan T. Stevenson. Algorithmic social engineering. *AEA Papers and Proceedings*, 110:96–100, May 2020. doi: 10.1257/pandp.20201037. URL https://www.aeaweb.org/articles?id=10.1257/pandp.20201037.

[15] Abir De, Nastaran Okati, Paramita Koley, Niloy Ganguly, and Manuel Gomez-Rodriguez. Regression under human assistance, 2021. URL https://arxiv.org/abs/1909.02963.

[16] Abir De, Nastaran Okati, Ali Zarezade, and Manuel Gomez-Rodriguez. Classification under human assistance, 2021. URL https://arxiv.org/abs/2006.11845.

[17] Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration, 2023. URL https://arxiv.org/abs/2305.13626.

[18] Kate Donahue, Sreenivas Gollapudi, and Kostas Kollias. When are two lists better than one?: Benefits and harms in joint decision-making, 2024. URL https://arxiv.org/abs/2308.11721.

[19] Yijiang River Dong, Tiancheng Hu, Zheng Hui, Caiqi Zhang, Ivan Vulić, Andreea Bobu, and Nigel Collier. Value of information: A framework for human-agent communication, 2026. URL https://arxiv.org/abs/2601.06407.

[20] 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, 2024. URL https://arxiv.org/abs/2307.01379.

[21] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010. URL http://jmlr.org/papers/v11/el-yaniv10a.html.

[22] Gheorghe Comanici et. al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2412.15115.

[23] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

[24] Qi Gao, Wei Xu, Hanxi Pan, Mowei Shen, and Zaifeng Gao. Human-centered human-ai collaboration (hchac), 2025. URL https://arxiv.org/abs/2505.22477.

[25] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks, 2017. URL https://arxiv.org/abs/1705.08500.

[26] Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option, 2019. URL https://arxiv.org/abs/1901.09192.

[27] Isaac Gibbs and Emmanuel Candès. Adaptive conformal inference under distribution shift, 2021. URL https://arxiv.org/abs/2106.00170.

[28] Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Towards uncertainty-aware language agent, 2024. URL https://arxiv.org/abs/2401.14016.

[29] Patrick Hemmer, Lukas Thede, Michael Vössing, Johannes Jakubik, and Niklas Kühl. Learning to defer with limited expert predictions, 2023. URL https://arxiv.org/abs/2304.07306.

[30] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

[31] Jessica Hullman, Yifan Wu, Dawei Xie, Ziyang Guo, and Andrew Gelman. Conformal prediction and human decision making, 2025. URL https://arxiv.org/abs/2503.11709.

[32] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity, 2024. URL https://arxiv.org/abs/2403.14403.

[33] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023. URL https://arxiv.org/abs/2305.06983.

[34] Sanyam Kapoor, Nate Gruver, Manley Roberts, Katherine Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew Gordon Wilson. Large language models must be taught to know what they don't know, 2025. URL https://arxiv.org/abs/2406.08391.

[35] Vijay Keswani, Matthew Lease, and Krishnaram Kenthapadi. Towards unbiased and accurate deferral to multiple experts, 2021. URL https://arxiv.org/abs/2102.13004.

[36] Shayan Kiyani, Sima Noorani, George Pappas, and Hamed Hassani. When to trust the cheap check: Weak and strong verification for reasoning, 2026. URL https://arxiv.org/abs/2602.17633.

[37] Klaus-Rudolf Kladny, Bernhard Schölkopf, and Michael Muehlebach. Conformal generative modeling with improved sample efficiency through sequential greedy filtering, 2025. URL https://arxiv.org/abs/2410.01660.

[38] Jon Kleinberg and Manish Raghavan. Algorithmic monoculture and social welfare. *Proceedings of the National Academy of Sciences*, 118(22), May 2021. ISSN 1091-6490. doi: 10.1073/pnas.2018340118. URL http://dx.doi.org/10.1073/pnas.2018340118.

[39] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Clam: Selective clarification for ambiguous questions with generative language models, 2023. URL https://arxiv.org/abs/2212.07769.

[40] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. URL https://arxiv.org/abs/2302.09664.

[41] 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.

[42] Belinda Z. Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models, 2023. URL https://arxiv.org/abs/2310.11589.

[43] Yibo Li, Miao Xiong, Jiaying Wu, and Bryan Hooi. Conftuner: Training large language models to express their confidence verbally, 2025. URL https://arxiv.org/abs/2508.18847.

[44] Linyu Liu, Yu Pan, Xiaocheng Li, and Guanting Chen. Uncertainty estimation and quantification for llms: A simple supervised approach, 2024. URL https://arxiv.org/abs/2404.15993.

[45] Terrance Liu and Zhiwei Steven Wu. Multi-group uncertainty quantification for long-form text generation, 2024. URL https://arxiv.org/abs/2407.21057.

[46] David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer, 2018. URL https://arxiv.org/abs/1711.06664.

[47] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models, 2023. URL https://arxiv.org/abs/2303.08896.

[48] Anqi Mao, Christopher Mohri, Mehryar Mohri, and Yutao Zhong. Two-stage learning to defer with multiple experts. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 3578–3606. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/0b17d256cf1fe1cc084922a8c6b565b7-Paper-Conference.pdf.

[49] Christopher Mohri and Tatsunori Hashimoto. Language models with conformal factuality guarantees, 2024. URL https://arxiv.org/abs/2402.10978.

[50] Yannis Montreuil, Shu Heng Yeo, Axel Carlier, Lai Xing Ng, and Wei Tsang Ooi. Optimal query allocation in extractive qa with llms: A learning-to-defer framework with theoretical guarantees, 2025. URL https://arxiv.org/abs/2410.15761.

[51] Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert, 2021. URL https://arxiv.org/abs/2006.01862.

[52] Sima Noorani, Shayan Kiyani, George Pappas, and Hamed Hassani. Human-ai collaborative uncertainty quantification. *arXiv preprint arXiv:2510.23476*, 2025.

[53] Sima Noorani, Shayan Kiyani, Hamed Hassani, and George Pappas. Multi-round human-ai collaboration with user-specified requirements. *arXiv preprint arXiv:2602.17646*, 2026.

[54] Nastaran Okati, Abir De, and Manuel Gomez-Rodriguez. Differentiable learning under triage, 2021. URL https://arxiv.org/abs/2103.08902.

[55] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2025. URL https://arxiv.org/abs/2406.18665.

[56] OpenAI. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

[57] William Overman and Mohsen Bayati. Conformal arbitrage: Risk-controlled balancing of competing objectives in language models, 2025. URL https://arxiv.org/abs/2506.00911.

[58] Edoardo Pona, Milad Kazemi, Mehran Hosseini, Yali Du, David Watson, Osvaldo Simeone, and Nicola Paoletti. Calibrate-then-delegate: Safety monitoring with risk and budget guarantees via model cascades, 2026. URL https://arxiv.org/abs/2604.14251.

[59] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs, 2018. URL https://arxiv.org/abs/1806.07011.

[60] Clara Punzi, Roberto Pellungrini, Mattia Setzu, Fosca Giannotti, and Dino Pedreschi. Learning paradigms for hybrid decision-making systems. *ACM Comput. Surv.*, April 2026. ISSN 0360-0300. doi: 10.1145/3802522. URL https://doi.org/10.1145/3802522. Just Accepted.

[61] Hongjin Qian and Zheng Liu. Scent of knowledge: Optimizing search-enhanced reasoning with information foraging. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=26kUrQm4zw.

[62] Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling, 2024. URL https://arxiv.org/abs/2306.10193.

[63] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[64] Maithra Raghu, Katy Blumer, Greg Corrado, Jon Kleinberg, Ziad Obermeyer, and Sendhil Mullainathan. The algorithmic automation problem: Prediction, triage, and human effort, 2019. URL https://arxiv.org/abs/1903.12220.

[65] Ramya Ramalingam, Shayan Kiyani, and Aaron Roth. The relationship between no-regret learning and online conformal prediction. *arXiv preprint arXiv:2502.10947*, 2025.

[66] Charvi Rastogi, Liu Leqi, Kenneth Holstein, and Hoda Heidari. A taxonomy of human and ml strengths in decision-making to investigate human-ml complementarity, 2023. URL https://arxiv.org/abs/2204.10806.

[67] 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, 2023. URL https://arxiv.org/abs/2307.01928.

[68] Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2call: When (not) to call tools, 2025. URL https://arxiv.org/abs/2504.18851.

[69] Maxon Rubin-Toles, Maya Gambhir, Keshav Ramji, Aaron Roth, and Surbhi Goel. Conformal language model reasoning with coherent factuality. In *The Thirteenth International Conference on Learning Representations*.

[70] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

[71] Hooman Shahrokhi, Devjeet Raj Roy, Yan Yan, Venera Arnaoudova, and Janaradhan Rao Doppa. Conformal prediction sets for deep generative models via reduction to conformal regression. *arXiv preprint arXiv:2503.10512*, 2025.

[72] Mark Steyvers, Heliodoro Tejeda, Gavin Kerrigan, and Padhraic Smyth. Bayesian modeling of human ai complementarity. *Proceedings of the National Academy of Sciences*, 119(11): e2111547119, 2022. doi: 10.1073/pnas.2111547119. URL https://www.pnas.org/doi/abs/10.1073/pnas.2111547119.

[73] Eleni Straitouri, Lequn Wang, Nastaran Okati, and Manuel Gomez Rodriguez. Improving expert predictions with conformal prediction, 2023. URL https://arxiv.org/abs/2201.12006.

[74] Eleni Straitouri, Suhas Thejaswi, and Manuel Gomez Rodriguez. Controlling counterfactual harm in decision support systems based on prediction sets, 2024. URL https://arxiv.org/abs/2406.06671.

[75] Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. Api is enough: Conformal prediction for large language models without logit-access, 2024. URL https://arxiv.org/abs/2403.01216.

[76] Sina Tayebati, Divake Kumar, Nastaran Darabi, Dinithi Jayasuriya, Ranganath Krishnan, and Amit Ranjan Trivedi. Learning conformal abstention policies for adaptive risk management in large language and vision-language models, 2025. URL https://arxiv.org/abs/2502.06884.

[77] Arsene Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, and Joumana Ghosn. Ddxplus: A new dataset for automatic medical diagnosis, 2022. URL https://arxiv.org/abs/2205.09148.

[78] Ran Tian, Masayoshi Tomizuka, Anca D. Dragan, and Andrea Bajcsy. Towards modeling and influencing the dynamics of human learning. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '23, page 350–358, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399647. doi: 10.1145/3568162.3578629. URL https://doi.org/10.1145/3568162.3578629.

[79] Giovanni De Toni, Nastaran Okati, Suhas Thejaswi, Eleni Straitouri, and Manuel Gomez-Rodriguez. Towards human-ai complementarity with prediction sets, 2024. URL https://arxiv.org/abs/2405.17544.

[80] Dennis Ulmer, Chrysoula Zerva, and André F. T. Martins. Non-exchangeable conformal language generation with nearest neighbors, 2024. URL https://arxiv.org/abs/2402.00707.

[81] Michael Vaccaro, Abdullah Almaatouq, and Thomas 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. URL https://doi.org/10.1038/s41562-024-02024-1.

[82] Rajeev Verma and Eric Nalisnick. Calibrated learning to defer with one-vs-all classifiers, 2022. URL https://arxiv.org/abs/2202.03673.

[83] Rajeev Verma, Daniel Barrejón, and Eric Nalisnick. Learning to defer to multiple experts: Consistent surrogate losses, confidence calibration, and conformal ensembles, 2023. URL https://arxiv.org/abs/2210.16955.

[84] Lequn Wang, Thorsten Joachims, and Manuel Gomez Rodriguez. Improving screening processes via calibrated subset selection. *ArXiv*, abs/2202.01147, 2022. URL https://api.semanticscholar.org/CorpusID:246473291.

[85] Wenxuan Wang, Juluan Shi, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen tse Huang, Wenxiang Jiao, and Michael R. Lyu. Learning to ask: When llm agents meet unclear instruction, 2026. URL https://arxiv.org/abs/2409.00557.

[86] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

[87] Bryan Wilder, Eric Horvitz, and Ece Kamar. Learning to complement humans. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165.

[88] Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your llm, 2024. URL https://arxiv.org/abs/2406.02543.

[89] Yasin Abbasi Yadkori, Ilja Kuzborskij, David Stutz, András György, Adam Fisch, Arnaud Doucet, Iuliya Beloshapka, Wei-Hung Weng, Yao-Yuan Yang, Csaba Szepesvári, Ali Taylan Cemgil, and Nenad Tomasev. Mitigating llm hallucinations via conformal abstention, 2024. URL https://arxiv.org/abs/2405.01563.

[90] Michael J. Zellinger, Rex Liu, and Matt Thomson. Cost-saving llm cascades with early abstention, 2025. URL https://arxiv.org/abs/2502.09054.

[91] Jiarui Zhang, Muzi Tao, Shangshang Wang, Ollie Liu, Xuezhe Ma, and Willie Neiswanger. An exam for active observers, 2026. URL https://arxiv.org/abs/2607.16165.

[92] Michael J. Q. Zhang, W. Bradley Knox, and Eunsol Choi. Modeling future conversation turns to teach llms to ask clarifying questions, 2025. URL https://arxiv.org/abs/2410.13788.

[93] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning, 2017. URL https://arxiv.org/abs/1709.00103.

Contents

A Extended Related Works **18**

B Proofs **19**

B.1 Proof of Theorem 3.1 19

B.2 Proof of Theorem 4.1 20

C Additional Experimental Results **23**

C.1 Support rate across all score variants 23

C.2 Ablation on the exploration probability μ 24

C.3 Sensitivity to the choice of g on VirtualHome 24

C.4 Scores histogram across the online stream 26

C.5 Per-task specifics and descriptive statistics 27

C.6 Error Control for all datasets 27

C.7 Hyperparameters 28

C.8 Task implementation details and example prompts 29

C.9 Comparison of models of varying strength 32

C.10 Experiments with Human Support on ActiveVision 33

C.11 End-to-End Task Performance and the Accuracy–Support Tradeoff 33

C.12 Operational Variants for the Score Input 35

A Extended Related Works

Training Agents to Seek Support. A significant body of work bakes support-seeking behavior into the agent itself through training, modifying the agent's weights or generation procedure so that it learns when to invoke external resources or ask clarifying questions. On the tool-use and retrieval side, Toolformer [70] self-supervisedly teaches a language model where to insert API calls by sampling and filtering candidate calls based on whether they reduce future-token loss. Self-RAG [5] trains a model to interleave retrieval with generation through reflection tokens that signal when to retrieve and when to commit. Confidence-triggered retrieval methods take a complementary route, deciding to retrieve based on signals during decoding: FLARE [33] triggers retrieval when the predicted next sentence contains low-confidence tokens, Adaptive-RAG [32] trains a small classifier to route queries among no-retrieval, single-step, and iterative-retrieval strategies, UALA [28] sets an uncertainty threshold on agent answers and resorts to external resources when exceeded, and SCENT [61] formalizes adaptive retrieval as reinforcement learning with dense rewards on intermediate retrieval quality. On the user-interaction side, GATE [42] introduced the framework of having an LLM elicit user preferences through targeted questions, with subsequent work training the questioner via self-improvement (STaR-GATE [2]), DPO-style preference optimization at the conversational-action level [92], and benchmarks that explicitly evaluate the ask-or-tool-call decision [68].

We instead provide an oversight layer around an unmodified agent, which decides whether to invoke support without changing how the agent itself generates outputs. This separation makes the two lines complementary rather than competing: any model that has already been trained to call tools, retrieve, or ask clarifying questions is precisely the kind of base policy our framework wraps. This enables us to deliver finite-sample error control on top of strong, already-trained agents without retraining, fine-tuning, or modifying generation, and to do so uniformly across support modalities. Our experiments use state-of-the-art LLMs (Qwen-2.5-7B, Gemini-2.5-Flash, GPT-4o-mini) that already incorporate substantial training of this kind, and our gains are reported on top of these strong agents.

Allocating Tasks Across Decision-Making Systems. A second body of work studies how to allocate tasks across multiple decision-making systems rather than how to support a single one. Within this body, four directions are most relevant. LLM cascades and routers send each query to the cheapest system whose output is good enough: FrugalGPT [12] composes LLM APIs into a cascade with learned stop criteria, RouteLLM [55] trains a router that selects between strong and weak LLMs using human preference data, and cost-saving cascades with early abstention [90] extend cascades to allow abstention at intermediate tiers.

Learning to defer [46] generalizes rejection learning by routing uncertain instances to an expert whose decision replaces the model's, with subsequent work developing Bayes-consistent surrogate losses [51], calibrated one-vs-all approaches [82], extensions to multiple experts [29, 35, 48, 83, 87], LLM allocation [50], algorithmic-triage formulations [15, 16, 54, 64], training-free conformal deferral [11], and collaborative matching for selective deferral [4]. Selective prediction lets the model abstain entirely on uncertain inputs, trading coverage for accuracy [21, 25, 26]. A more recent line uses conformal calibration to deliver formal guarantees on these decisions: Conformal Arbitrage [57] calibrates a threshold mediating between a primary model and a more conservative guardian, and Calibrate-Then-Delegate (CTD) [58] introduces a delegation-value probe that predicts the benefit of escalating to an expert and calibrates a threshold on this probe via offline multiple-hypothesis testing.

We instead study when an unmodified agent should invoke support to augment its own reasoning, rather than which of two systems should produce the output. Although we study a different problem, the combination of distribution-free online calibration, importance-weighted threshold updates under counterfactual partial feedback, and control of a conditional miss-rate, together with the algorithmic techniques we develop may be of independent interest to the broader umbrella L2D and selective classification literatures.

Inference-Time Policies for Support-Seeking. A closely related line of work designs inference-time policies that decide whether an agent should seek information before acting. CLAM [39] prompts an LLM to classify whether a user question is ambiguous and asks a clarifying question when it is. AwN [85] prompts the LLM to ask whenever it encounters obstacles caused by unclear instructions during tool use. Earlier proactive-dialogue work uses prompting to elicit clarifying

behavior more generally [17]. More recent work introduces formal objectives: KnowNo [67] uses split conformal prediction on a held-out calibration set to construct prediction sets over candidate user intents, asking for help whenever the set is non-singleton; the Value-of-Information policy of Dong et al. [19] maintains a belief distribution over latent intents, generates candidate clarifying questions, simulates user replies, and asks when the expected utility gain exceeds an explicit asking cost. A complementary perspective is taken by Kiyani et al. [36], who frame support-seeking on the *verification* side rather than the generation side: their weak–strong verification policies decide when a cheap internal check (e.g., self-consistency, a proxy reward) suffices and when to defer to costly external verification (e.g., user inspection), with an online algorithm that provably controls both acceptance and rejection errors.

We instead provide a unifying framework that brings these approaches under the same design principles, through an oversight layer with rigorous finite-sample error control. This enables us to handle a broad set of support modalities — clarifying questions, external tools, additional evidence, expert guidance — within a single algorithm, to operate fully online without an offline calibration set, and to control a counterfactual conditional error at a user-chosen level. Our experiments target precisely these capabilities: covering information gathering, tool use, and human-AI collaboration under the same algorithm, with online error control across a range of target levels.

AI as Decision Support System for humans A long line of work studies how AI systems should support human decision-makers. One strand uses prediction sets as a structured interface for collaboration, calibrating the set to improve human accuracy [6, 8, 73, 79, 84], analyzing decision-relevant uncertainty [31], and formalizing counterfactual harm from set-based support [52, 53, 74]. A second strand develops theoretical frameworks for human-AI complementarity, identifying when joint systems surpass either agent alone [7, 66, 72, 81] and how algorithmic outputs shape human choices [9, 10, 14, 18, 24, 38, 78]. recent surveys taxonomize the broader space of human-machine hybrid decision-making paradigms [60].

In all of these, the human is the final decision-maker and the AI provides support around them. Our work reverses this configuration: the AI agent is the decision-maker, and humans, tools, and auxiliary information serve as support mechanisms around it.

Uncertainty quantification for AI Agents A separate literature studies how to elicit reliable uncertainty estimates from LLMs more generally. Conformal prediction methods construct prediction sets with formal coverage guarantees over open-ended LLM outputs [13, 37, 41, 45, 49, 62, 69, 71, 75, 80]; a complementary line uses conformal risk control to learn abstention or selective-prediction policies [76, 88, 89]. A broader thread mitigates hallucinations through direct uncertainty estimation, semantic-disagreement signals across sampled responses, or verifier-based detectors [1, 20, 23, 40, 44, 47, 86], and recent work fine-tunes LLMs for better-calibrated verbalized confidence [34, 43]. These methods produce signals about the agent's confidence in its own output. Better LLM uncertainty estimators from this line could be used as the anchor in our anchored score, providing a stronger initial signal that calibration-on-the-fly need only refine.

B Proofs

B.1 Proof of Theorem 3.1

Proof. Let

$$G := g(X, Y_0, Y_1), \qquad \pi_1 := \mathbb{P}(G = 1),$$

and assume $\pi_1 > 0$. For any measurable strategy $a : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, the constraint

$$\mathbb{P}(a(X, Y_0) = 0 \mid G = 1) \le \varepsilon$$

is equivalent to

$$\mathbb{P}(a(X, Y_0) = 1 \mid G = 1) \ge 1 - \varepsilon.$$

By the definition

$$\mathrm{val}(x, y_0) := \mathbb{P}(G = 1 \mid X = x, Y_0 = y_0),$$

we have

$$\mathbb{P}(a(X, Y_0) = 1 \mid G = 1) = \frac{\mathbb{E}[a(X, Y_0)\mathbf{1}\{G = 1\}]}{\pi_1} = \frac{\mathbb{E}[a(X, Y_0)\,\mathrm{val}(X, Y_0)]}{\pi_1},$$

where the second equality follows from the tower property. Hence (SDS-Opt) is equivalent to

$$\min_{a:\mathcal{X}\times\mathcal{Y}\to\{0,1\}} \mathbb{E}[a(X,Y_0)] \qquad \text{subject to} \qquad \mathbb{E}[a(X,Y_0)\,\mathrm{val}(X,Y_0)] \geq (1-\varepsilon)\pi_1.$$

Let

$$\beta := (1-\varepsilon)\pi_1.$$

The objective charges one unit for every point on which support is sought, while the constraint credits such a point in proportion to $\mathrm{val}(X,Y_0)$. Therefore, an optimal strategy should allocate support to the largest values of val first. We now make this formal.

Choose a threshold $\tau^\star \in [0,1]$ such that

$$\mathbb{E}[\mathrm{val}(X,Y_0)\mathbf{1}\{\mathrm{val}(X,Y_0) > \tau^\star\}] \leq \beta \leq \mathbb{E}[\mathrm{val}(X,Y_0)\mathbf{1}\{\mathrm{val}(X,Y_0) \geq \tau^\star\}]\,.$$

If necessary, randomize on the boundary $\{\mathrm{val}(X,Y_0) = \tau^\star\}$ with probability $\rho^\star \in [0,1]$ chosen so that

$$\mathbb{E}\Big[\mathrm{val}(X,Y_0)\Big(\mathbf{1}\{\mathrm{val}(X,Y_0) > \tau^\star\} + \rho^\star\mathbf{1}\{\mathrm{val}(X,Y_0) = \tau^\star\}\Big)\Big] = \beta.$$

Define

$$a^\star(x,y_0) = \mathbf{1}\{\mathrm{val}(x,y_0) > \tau^\star\} + \rho^\star\mathbf{1}\{\mathrm{val}(x,y_0) = \tau^\star\}.$$

Equivalently, $a^\star$ seeks support whenever $\mathrm{val}(x,y_0)$ is above the threshold and randomizes on the boundary if needed. By construction, $a^\star$ satisfies the constraint.

It remains to show optimality. Let a be any feasible strategy, allowing randomized values in $[0,1]$; this only enlarges the feasible class. By the definition of $a^\star$, we have pointwise

$$(a(x,y_0) - a^\star(x,y_0))(\mathrm{val}(x,y_0) - \tau^\star) \leq 0.$$

Taking expectations gives

$$\mathbb{E}[(a-a^\star)\,\mathrm{val}] \leq \tau^\star\mathbb{E}[a-a^\star].$$

Since a is feasible and $a^\star$ satisfies the constraint with equality,

$$\mathbb{E}[(a-a^\star)\,\mathrm{val}] \geq 0.$$

Therefore,

$$0 \leq \tau^\star\mathbb{E}[a-a^\star].$$

If $\tau^\star > 0$, this implies

$$\mathbb{E}[a] \geq \mathbb{E}[a^\star].$$

If $\tau^\star = 0$, feasibility requires $a = 1$ almost surely on the set where $\mathrm{val}(X,Y_0) > 0$, and $a^\star$ is the minimal such rule, so the same conclusion holds. Thus no feasible strategy uses support less often than $a^\star$.

Hence an optimal solution is given by a threshold rule in $\mathrm{val}(x,y_0)$, with possible randomization on the boundary. This proves the claim. □

B.2 Proof of Theorem 4.1

Proof. Fix a horizon T. For the analysis, define

$$g_t := g(x_t, y_0^t, y_1^t) \in \{0,1\}$$

as the potential benefit of support on round t, whether or not support is actually sought. All probabilities below are over the algorithm's internal Bernoulli draws. If

$$N_g(T) := \sum_{t=1}^{T} g_t = 0,$$

then $\widehat{\mathrm{MSE}}(T) = 0$ by convention, and the claim is immediate. Hence assume

$$N := N_g(T) = \sum_{t=1}^{T} g_t \geq 1.$$

We first outline the proof. The threshold update controls a threshold-induced version of the missed-support error, in which the realized missed-support indicator $(1 - a_t)$ is replaced by its conditional mean $(1 - p_t)\mathbf{1}\{s_t < \lambda_t\}$. The update gives a telescoping identity for this quantity, up to an importance-weighted martingale error. We control this martingale error by Freedman's inequality. Finally, we compare the threshold-induced error to the actual empirical error $\widehat{\mathrm{MSE}}(T)$ using a second martingale concentration bound.

Claim 1. Define

$$\overline{\mathrm{MSE}}(T) := \frac{1}{N}\sum_{t=1}^{T} g_t(1 - p_t)\mathbf{1}\{s_t < \lambda_t\}.$$

Then, with probability at least $1 - \delta/2$,

$$\overline{\mathrm{MSE}}(T) \le \alpha + \frac{1 + 2\eta/\mu}{\eta N} + \sqrt{\frac{2\log(4/\delta)}{\mu N}} + \frac{\log(4/\delta)}{3\mu N}.$$

Proof of Claim 1. Define

$$\bar{e}_t := g_t\Big((1 - p_t)\mathbf{1}\{s_t < \lambda_t\} - \alpha\Big), \qquad \hat{e}_t := \frac{g_t a_t}{p_t}\Big((1 - p_t)\mathbf{1}\{s_t < \lambda_t\} - \alpha\Big).$$

The threshold update can be written as

$$\lambda_{t+1} = \lambda_t - \eta\hat{e}_t.$$

Therefore,

$$\sum_{t=1}^{T} \hat{e}_t = \frac{\lambda_1 - \lambda_{T+1}}{\eta}.$$

Moreover,

$$\sum_{t=1}^{T} \bar{e}_t = N\big(\overline{\mathrm{MSE}}(T) - \alpha\big).$$

Let $\mathcal{F}_{t-1}$ denote the history before round t, and define

$$\mathcal{G}_t := \sigma(\mathcal{F}_{t-1}, x_t, y_0^t, s_t, \lambda_t, p_t, g_t).$$

This filtration contains all pre-action quantities at round t, together with the potential benefit variable g_t used only for the analysis. Conditional on $\mathcal{G}_t$, the only randomness in round t is

$$a_t \sim \mathrm{Bernoulli}(p_t).$$

Hence

$$\mathbb{E}[\hat{e}_t \mid \mathcal{G}_t] = g_t\Big((1 - p_t)\mathbf{1}\{s_t < \lambda_t\} - \alpha\Big)\mathbb{E}\left[\frac{a_t}{p_t}\,\middle|\,\mathcal{G}_t\right] = \bar{e}_t.$$

Thus, with

$$Z_t := \hat{e}_t - \bar{e}_t,$$

the sequence $(Z_t)_{t\ge1}$ is a martingale difference sequence, and

$$N\big(\overline{\mathrm{MSE}}(T) - \alpha\big) = \frac{\lambda_1 - \lambda_{T+1}}{\eta} - \sum_{t=1}^{T} Z_t.$$

We next bound the threshold sequence. Since $p_t \ge \mu$ and

$$\Big|(1 - p_t)\mathbf{1}\{s_t < \lambda_t\} - \alpha\Big| \le 1,$$

we have

$$|\hat{e}_t| \le \frac{1}{\mu}.$$

If $\lambda_t < 0$, then $s_t \geq \lambda_t$ because $s_t \in [0,1]$, so $p_t = 1$ and $\mathbf{1}\{s_t < \lambda_t\} = 0$. Hence

$$\hat{e}_t = -\alpha g_t a_t \leq 0,$$

and therefore $\lambda_{t+1} \geq \lambda_t$. If $\lambda_t > 1$, then $s_t < \lambda_t$, so $p_t = \mu$ and $\mathbf{1}\{s_t < \lambda_t\} = 1$. Since $\alpha < 1 - \mu$,

$$\hat{e}_t = \frac{g_t a_t}{\mu}\big((1-\mu) - \alpha\big) \geq 0,$$

and therefore $\lambda_{t+1} \leq \lambda_t$. Starting from $\lambda_1 \in [0,1]$, these facts imply by induction that

$$\lambda_t \in \left[-\frac{\eta}{\mu},\, 1 + \frac{\eta}{\mu}\right] \qquad \text{for all } t \leq T+1.$$

Consequently,

$$|\lambda_{T+1} - \lambda_1| \leq 1 + \frac{2\eta}{\mu}.$$

It remains to control the martingale term. Write

$$Z_t = c_t\left(\frac{a_t}{p_t} - 1\right), \qquad c_t := g_t\Big((1-p_t)\mathbf{1}\{s_t < \lambda_t\} - \alpha\Big).$$

Since $|c_t| \leq 1$ and $p_t \geq \mu$,

$$|Z_t| \leq \frac{1}{\mu}.$$

Furthermore,

$$\mathrm{Var}(Z_t \mid \mathcal{G}_t) = c_t^2 \,\mathrm{Var}\left(\frac{a_t}{p_t} \,\middle|\, \mathcal{G}_t\right).$$

Because $a_t \sim \mathrm{Bernoulli}(p_t)$,

$$\mathrm{Var}\left(\frac{a_t}{p_t} \,\middle|\, \mathcal{G}_t\right) = \frac{1-p_t}{p_t} \leq \frac{1}{\mu}.$$

Also $c_t^2 \leq g_t$, since $g_t \in \{0,1\}$. Therefore,

$$\mathrm{Var}(Z_t \mid \mathcal{G}_t) \leq \frac{g_t}{\mu}.$$

Thus the predictable quadratic variation satisfies

$$\sum_{t=1}^{T} \mathrm{Var}(Z_t \mid \mathcal{G}_t) \leq \frac{N}{\mu}.$$

By Freedman's inequality, with probability at least $1 - \delta/2$,

$$\left|\sum_{t=1}^{T} Z_t\right| \leq \sqrt{\frac{2N\log(4/\delta)}{\mu}} + \frac{\log(4/\delta)}{3\mu}.$$

Combining the preceding displays gives

$$\overline{\mathrm{MSE}}(T) \leq \alpha + \frac{1 + 2\eta/\mu}{\eta N} + \sqrt{\frac{2\log(4/\delta)}{\mu N}} + \frac{\log(4/\delta)}{3\mu N}.$$

This proves Claim 1. ◇

Claim 2. With probability at least $1 - \delta/2$,

$$\widehat{\mathrm{MSE}}(T) \leq \overline{\mathrm{MSE}}(T) + \sqrt{\frac{2\log(4/\delta)}{N}} + \frac{\log(4/\delta)}{3N}.$$

Proof of Claim 2. By definition,

$$\widehat{\mathrm{MSE}}(T) = \frac{1}{N}\sum_{t=1}^{T} g_t(1 - a_t).$$

Since $p_t = 1$ whenever $s_t \geq \lambda_t$, we have $a_t = 1$ almost surely on those rounds. Therefore,

$$\widehat{\mathrm{MSE}}(T) = \frac{1}{N} \sum_{t=1}^{T} g_t (1 - a_t) \mathbf{1}\{s_t < \lambda_t\}.$$

Define

$$U_t := g_t \mathbf{1}\{s_t < \lambda_t\} \Big((1 - a_t) - (1 - p_t) \Big) = g_t \mathbf{1}\{s_t < \lambda_t\} (p_t - a_t).$$

Then

$$\widehat{\mathrm{MSE}}(T) - \overline{\mathrm{MSE}}(T) = \frac{1}{N} \sum_{t=1}^{T} U_t.$$

Moreover,

$$\mathbb{E}[U_t \mid \mathcal{G}_t] = 0,$$

so $(U_t)_{t \geq 1}$ is a martingale difference sequence. Since $|U_t| \leq 1$ and

$$\mathrm{Var}(U_t \mid \mathcal{G}_t) = g_t \mathbf{1}\{s_t < \lambda_t\} \, \mathrm{Var}(a_t \mid \mathcal{G}_t) = g_t \mathbf{1}\{s_t < \lambda_t\} p_t (1 - p_t) \leq g_t,$$

the predictable quadratic variation is at most N. Freedman's inequality gives, with probability at least $1 - \delta/2$,

$$\left| \sum_{t=1}^{T} U_t \right| \leq \sqrt{2N \log(4/\delta)} + \frac{\log(4/\delta)}{3}.$$

Dividing by N proves Claim 2. ◇

Combining Claims 1 and 2 with a union bound, with probability at least $1 - \delta$,

$$\widehat{\mathrm{MSE}}(T) \leq \alpha + \frac{1 + 2\eta/\mu}{\eta N} + \sqrt{\frac{2 \log(4/\delta)}{\mu N}} + \frac{\log(4/\delta)}{3\mu N} + \sqrt{\frac{2 \log(4/\delta)}{N}} + \frac{\log(4/\delta)}{3N}.$$

Since $\mu \leq 1$,

$$\sqrt{\frac{2 \log(4/\delta)}{N}} \leq \sqrt{\frac{2 \log(4/\delta)}{\mu N}}, \qquad \frac{\log(4/\delta)}{3N} \leq \frac{\log(4/\delta)}{3\mu N}.$$

Therefore,

$$\widehat{\mathrm{MSE}}(T) \leq \alpha + \frac{1 + 2\eta/\mu}{\eta N} + 2\sqrt{\frac{2 \log(4/\delta)}{\mu N}} + \frac{2 \log(4/\delta)}{3\mu N}.$$

This is slightly stronger than the stated bound. Since

$$2\sqrt{\frac{2 \log(4/\delta)}{\mu N}} = \sqrt{\frac{8 \log(4/\delta)}{\mu N}},$$

and

$$\frac{2 \log(4/\delta)}{3\mu N} \leq \frac{4 \log(4/\delta)}{3\mu N},$$

we obtain

$$\widehat{\mathrm{MSE}}(T) \leq \alpha + \frac{1 + 2\eta/\mu}{\eta N} + \sqrt{\frac{8 \log(4/\delta)}{\mu N}} + \frac{4 \log(4/\delta)}{3\mu N}.$$

Recalling that $N = N_g(T)$ proves the theorem. □

C Additional Experimental Results

C.1 Support rate across all score variants

Figure 3 in the main text shows the cumulative support rate for the best-performing variant in each of the Anchored and Representation families, together with its counterpart in the other family. Figure 5 reports the same quantity for every score variant we consider, providing the full picture of how each embedding choice and family performs at matched missed-support error.

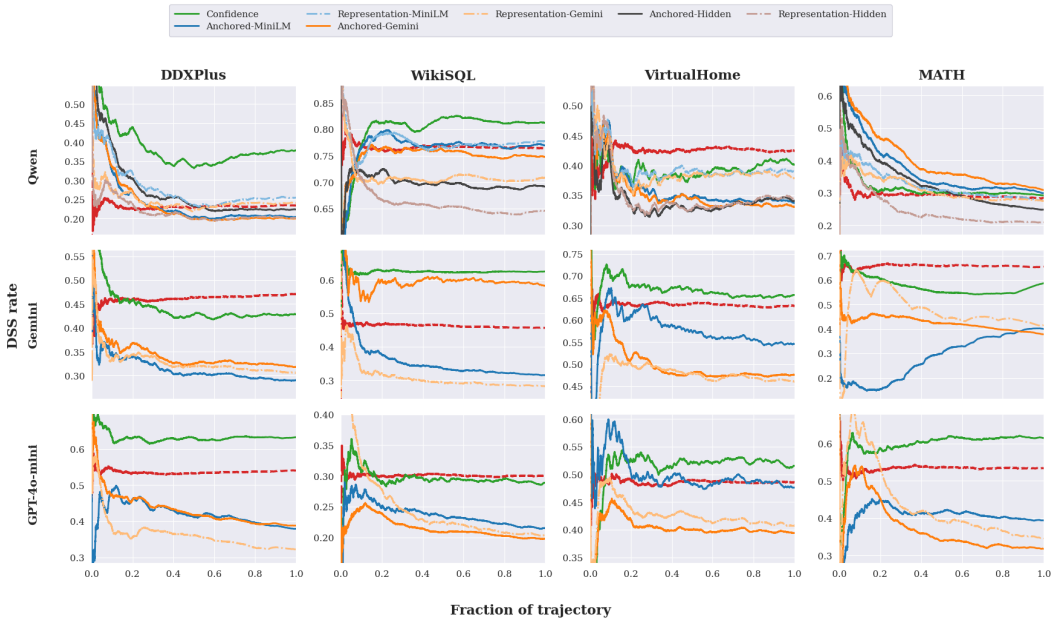

Figure 5: Cumulative support rate $\widehat{\mathrm{SR}}_T = \frac{1}{T}\sum_{t=1}^{T} a_t$ across all task–model pairs, showing every score variant. Rows are base agents, columns are tasks. All variants are run at the same target α as in Figure 3.

C.2 Ablation on the exploration probability μ

Theorem 4.1 identifies the exploration probability μ as the lever that controls the second slack term in the missed-support error bound. larger values of μ tighten error control and yield smoother convergence, but increase support usage, since the algorithm calls support with probability μ on every round where the score falls below the threshold. Here we verify this empirically by sweeping μ on a fixed task–model–score configuration and observing how the missed-support error and the support rate move together.

We fix the base agent to GPT-4o-mini, the task to DDXPlus, and the score to Anchored-Gemini. The threshold step size η_t and initial threshold λ_0 are held constant and only μ is varied. The configurations are listed in Table C.2.

Figure 6 reports the cumulative missed-support error and the cumulative support rate across the five values of μ. The two panels show the predicted tradeoff. On the error-control side, larger values of μ produce smoother trajectories that converge faster to the target α, while small values exhibit noisier adaptation. On the support-efficiency side, small μ delivers substantially lower support rates, while large μ pushes the support rate up because the algorithm queries support with non-negligible probability even on rounds where the score is decisively low. In our main experiments we therefore choose moderate values of μ that achieve reliable error control without sacrificing the support-rate gains.

C.3 Sensitivity to the choice of g on VirtualHome

The benefit function g encodes what it means for support to materially help on a given task, and its definition naturally depends on the application and on what the provider considers a meaningful improvement. The purpose of this section is to verify that our oversight layer is agnostic to this choice: error control holds for any g, and the qualitative ranking of score variants is preserved across different definitions.

We illustrate this on VirtualHome, which is the most since the output is a structured plan rather than a verifiable answer. Each generated plan is evaluated against the gold plan using its longest

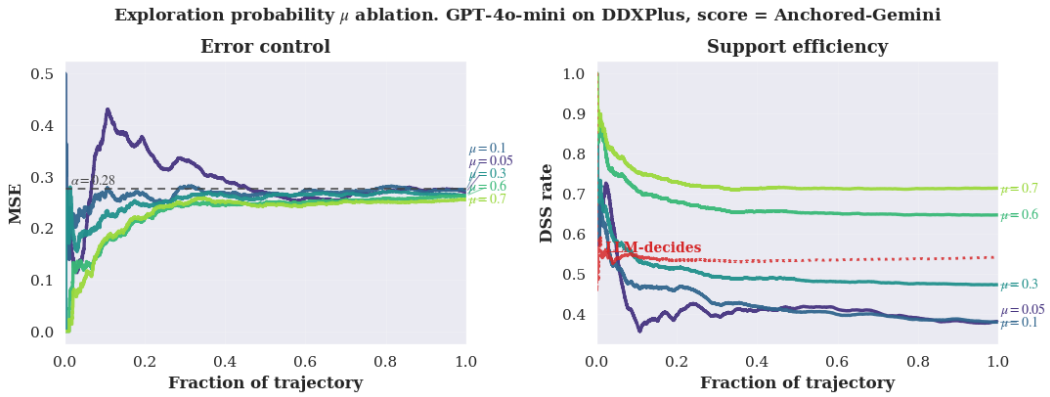

Figure 6: Effect of the exploration probability μ. Base agent: GPT-4o-mini. Task: DDXPlus. Score: Anchored-Gemini. Left: cumulative missed-support error against the target α. Right: cumulative support rate, with the LLM-DECIDES baseline shown for reference. Larger μ tightens error control and yields smoother convergence but increases support usage, matching the dependence on μ in the slack term of Theorem 4.1.

μ	η_λ	λ_0
0.05	0.015	0.5
0.10	0.015	0.5
0.30	0.015	0.5
0.60	0.015	0.5
0.70	0.015	0.5

Table 1: Hyperparameters for the exploration-probability ablation. Base agent: GPT-4o-mini. Task: DDXPlus. Score: Anchored-Gemini. μ is varied while η_t and λ_0 are held at the base values from the main experiments.

common subsequence (LCS) score, which measures how much of the gold plan's ordered structure the generated plan recovers. We compare two natural gain definitions built on top of this score:

- g_{strict} marks support as beneficial when the supported plan reaches an absolute quality bar, $y_1^{\text{LCS}} \geq 0.5$. This captures the view that improvement matters only if the supported plan crosses a fixed threshold of acceptability.
- g_{improved} marks support as beneficial whenever the supported plan strictly improves on the unsupported plan in LCS score. This is the most permissive definition and treats any strict improvement as material.

The two definitions impose qualitatively different demands. The first asks whether y_1 is good in absolute terms, the second asks whether it is better than y_0. They induce different positive rates and therefore different targets: the missed-support rate of LLM-DECIDES is $\alpha = 0.51$ under g_{strict} and $\alpha = 0.34$ under g_{improved}, and we set the target accordingly so all comparisons remain at matched error.

Figure 7 reports both the cumulative missed-support error (top row) and the cumulative support rate (bottom row) for Gemini-2.5-Flash on VirtualHome under both gain definitions. Two observations stand out. First, error control holds in every panel. The running missed-support error converges to the corresponding target α regardless of which g is used, consistent with Theorem 4.1. Second, the qualitative ranking of methods carries over: the adaptive parameterized score variants reduce the support rate substantially relative to LLM-DECIDES. This confirms that g is a flexible engineering choice the practitioner can tailor to their task and their notion of meaningful improvement, without changing the algorithm or its guarantees.



Figure 7: Gemini-2.5-Flash on VirtualHome under two gain definitions. Columns are gain definitions: g_{strict} (y_1 reaches LCS ≥ 0.5) on the left, g_{improved} (y_1 improves over y_0 in LCS) on the right. Top row reports the cumulative missed-support error against the corresponding target α; bottom row reports the cumulative support rate. Across both definitions, the running error converges to α and the parameterized variants reduce support usage relative to LLM-DECIDES.

C.4 Scores histogram across the online stream

To complement the support-rate results in Section 5, we visualize the score distributions produced by each score family along the online stream. Figure 8 shows histograms of the score s split by the latent benefit variable g for Gemini-2.5-Flash on all four tasks. The three rows correspond to the three score families introduced in Section 5.1: the raw LLM confidence $\hat{g}_{\text{LLM}}$, the Anchored score that adds a calibrated residual on top of $\hat{g}_{\text{LLM}}$, and the Representation score that drops the anchor and learns from the embedding alone. The Anchored and Representation variants both use the GEMINI-EMBEDDING-2 model (ϕ = Gemini) and are calibrated on the fly.

The top row makes the limitation of relying on raw confidence concrete. On WikiSQL and MATH, both $g = 0$ and $g = 1$ pile up at the similar scores, so LLM Confidence carries almost no information about whether support would help; on DDXPlus and VirtualHome the two distributions overlap heavily as well. A threshold rule applied directly to this score is therefore forced to either accept too many $g = 0$ instances or call support on too many $g = 1$ instances, which is why Confidence ends up close to or above LLM-DECIDES in support efficiency.

The middle and bottom rows show what calibration-on-the-fly recovers from the same stream. The Anchored score sharpens the confidence signal where it carries useful information (DDXPlus, VirtualHome) and corrects it where it does not (WikiSQL, MATH), pushing the $g = 1$ mass toward high scores and the $g = 0$ mass toward low scores. The Representation score, which has no anchor to start from, learns this separation directly from the embedding. This is the mechanism that makes the parameterized score families consistently outperform Confidence in Figure 4. Even when the underlying signal is uninformative or misleading, the calibration-on-the-fly extracts a usable separation between beneficial and non-beneficial instances from the feedback received on rounds where support is invoked.

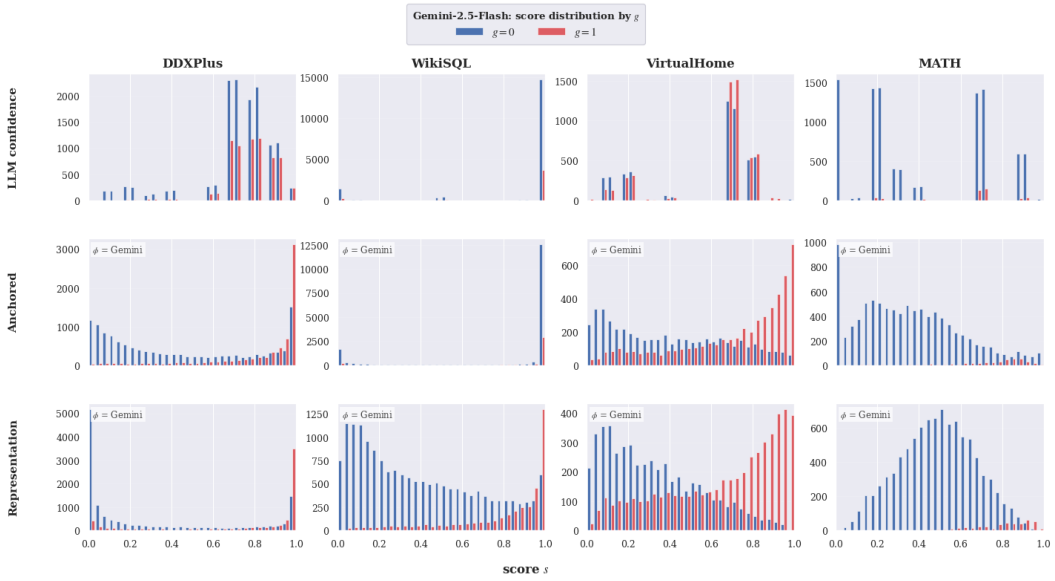

Figure 8: Score distributions along the online stream for Gemini-2.5-Flash, split by the latent benefit variable g. Rows are score families (LLM confidence, Anchored, Representation) and columns are tasks (DDXPlus, WikiSQL, VirtualHome, MATH). Anchored and Representation variants both use the Gemini embedding ($\phi = \text{Gemini}$) and are calibrated on the fly. Raw confidence often fails to separate $g = 0$ from $g = 1$, particularly on WikiSQL and MATH where both classes concentrate at the same scores. After calibration-on-the-fly, the Anchored and Representation scores recover separation between the two classes, providing the threshold rule with a more informative signal.

C.5 Per-task specifics and descriptive statistics

Table 2 reports descriptive statistics for each task–model pair. We include the accuracy of the unsupported output y_0, the accuracy of the supported output y_1, the empirical probability $\Pr(g = 1)$ that support is beneficial, and the support rate and missed-support error of the LLM-DECIDES baseline. These quantities highlight that the value of support depends jointly on the task and the base model: even within the same task, support can be far more useful for one model than another. For example, on DDXPlus, Gemini-2.5-Flash benefits substantially more from the additional clinical evidence than Qwen-2.5-7B, despite both receiving identical support. The headroom for support to help is therefore a property of the task, model, and support modality together, rather than of the task alone. This variability is precisely what motivates the oversight layer we develop in the paper. Rather than relying on the agent's own support-seeking behavior, which can drift far from $\Pr(g = 1)$ in either direction, our algorithm adapts online to the specific task–model–support combination at hand and controls the missed-support error at a user-chosen level.

C.6 Error Control for all datasets

Figure 9 reports the cumulative missed-support error $\mathrm{MSE}(T)$ across the full 3×4 grid of base agents and tasks. The main text shows convergence for a single representative pair (Figure 3); here we verify that the same behavior holds in every panel. Across all twelve task–model pairs and across every score variant, the running MSE converges to the target level α, set to the rate achieved by LLM-DECIDES on that pair. This confirms that the finite-sample guarantee of Theorem 4.1 holds uniformly across tasks, models, and score families. The bound in that theorem decomposes into two qualitatively different sources of slack. The first is the intrinsic error of online quantile tracking, governed by the threshold step size η: larger values facilitate faster initial convergence but produce noisier trajectories, while smaller values yield smoother curves at the cost of slower adaptation. The second arises from partial feedback and the randomized exploration needed to obtain unbiased

Model	Task	y_0 accuracy	y_1 accuracy	$\Pr(g = 1)$	LLM DSS rate	LLM MSE
Qwen-2.5-7B	DDXPlus	37.4%	36.9%	0.106	0.234	0.708
	WikiSQL	60.2%	74.4%	0.240	0.764	0.108
	VirtualHome	8.4%	9.6%	0.416	0.424	0.577
	MATH	57.4%	66.6%	0.168	0.284	0.714
Gemini-2.5-Flash	DDXPlus	49.8%	76.5%	0.341	0.471	0.528
	WikiSQL	63.4%	76.5%	0.190	0.457	0.425
	VirtualHome	13.6%	20.0%	0.514	0.632	0.342
	MATH	90.8%	92.2%	0.042	0.654	0.143
GPT-4o-mini	DDXPlus	44.1%	54.7%	0.203	0.540	0.276
	WikiSQL	70.8%	79.3%	0.167	0.300	0.633
	VirtualHome	15.0%	25.8%	0.614	0.486	0.463
	MATH	60.6%	68.4%	0.130	0.534	0.369

Table 2: Per (task,model) descriptive statistics for each base agent.

feedback, and is controlled by the exploration probability μ: larger μ tightens error control but increases support usage. We study the dependence on μ empirically in Appendix C.2.

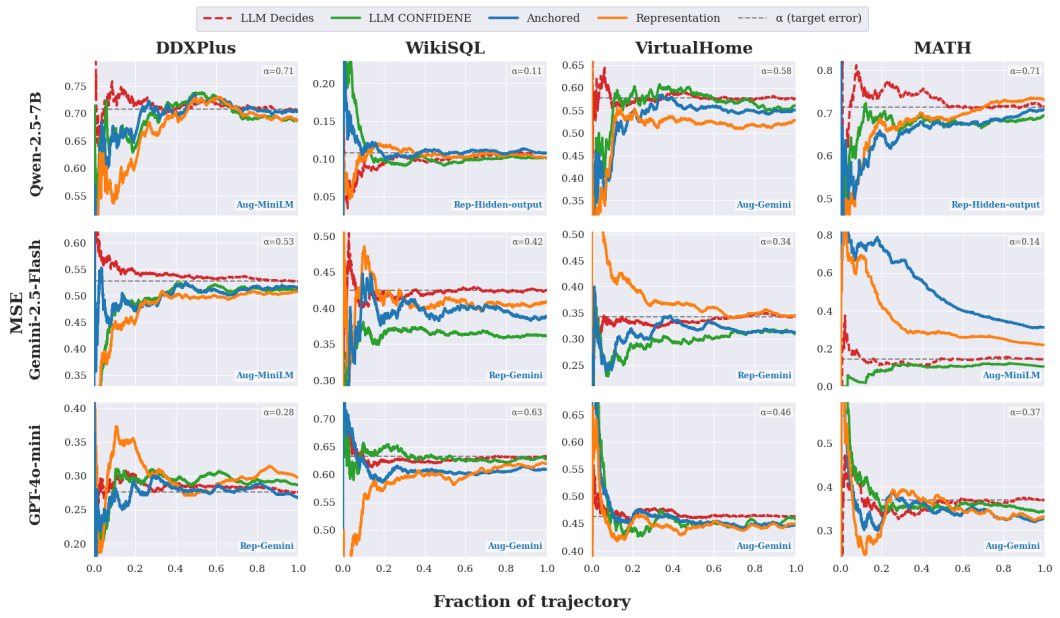

Figure 9: Cumulative missed-support error $\mathrm{MSE}(T)$ across all task–model pairs. Rows are base agents (Qwen-2.5-7B, Gemini-2.5-Flash, GPT-4o-mini), columns are tasks (DDXPlus, WikiSQL, VirtualHome, MATH). Each panel shows the running MSE for all score variants together with the LLM-DECIDES baseline and the target level α. All variants converge towards α regardless of the score family.

C.7 Hyperparameters

For completeness, Table 3 reports the full hyperparameter configuration used to produce the main-text results across all base agents, tasks, and score variants. We report the calibration step size η_t, the threshold step size γ_t, the exploration probability μ, and the initial threshold λ_0. Confidence has no learnable parameters and so its η_t entry is left blank. Hidden-state variants are only available in the

white-box regime and are reported for Qwen-2.5-7B; the corresponding rows for Gemini-2.5-Flash and GPT-4o-mini are left blank.

The values reported in Table 3 were selected by a simple grid search over a small set of candidate values on a held out test stream for each hyperparameter, with the support rate at the end of the trajectory used as the selection criterion. We did not find this search to be delicate. Across all task–model pairs, a wide range of reasonable choices yield similar performance, and the qualitative conclusions of the paper are insensitive to the exact configuration. In practice, the threshold step size η_t governs how quickly the algorithm adapts; the exploration probability μ trades off tighter error control against higher support usage and is studied directly in Appendix C.2; and the score update step size γ_t and initial threshold λ_0 have only secondary effects once the other two are set sensibly. We therefore expect a practitioner to obtain comparable results without an extensive hyperparameter search.

Model	Variant	DDXPlus				WikiSQL				VirtualHome				MATH			
		γ_t	η_t	μ	λ_0	γ_t	η_t	μ	λ_0	γ_t	η_t	μ	λ_0	γ_t	η_t	μ	λ_0
Qwen-2.5-7B	Confidence	—	0.05	0.1	0.4	—	0.01	0.2	0.3	—	0.05	0.1	0.4	—	0.1	0.25	0.6
	Anchored-MiniLM	2	0.05	0.1	0.5	2	0.01	0.2	0.3	2	0.05	0.1	0.5	2	0.01	0.1	0.6
	Representation-MiniLM	10	0.05	0.1	0.5	2	0.01	0.2	0.3	10	0.05	0.1	0.5	2	0.01	0.1	0.6
	Anchored-Gemini	2	0.05	0.1	0.5	2	0.01	0.2	0.3	2	0.05	0.1	0.5	2	0.01	0.1	0.6
	Representation-Gemini	5	0.05	0.1	0.5	2	0.01	0.2	0.3	10	0.05	0.1	0.5	2	0.01	0.1	0.6
	Anchored-Hidden	2	0.05	0.1	0.5	2	0.01	0.2	0.3	0.5	0.05	0.1	0.5	5	0.01	0.1	0.6
	Representation-Hidden	2	0.05	0.1	0.5	2	0.01	0.2	0.3	2	0.05	0.1	0.5	5	0.01	0.1	0.6
Gemini-2.5-Flash	Confidence	—	0.01	0.1	0.5	—	0.001	0.3	0.99	—	0.01	0.05	0.6	—	0.01	0.1	0.2
	Anchored-MiniLM	2	0.05	0.1	0.5	2	0.05	0.1	0.5	1	0.05	0.05	0.5	1	0.01	0.1	0.7
	Representation-MiniLM	2	0.01	0.2	0.3	2	0.01	0.2	0.3	2	0.01	0.2	0.3	2	0.01	0.2	0.3
	Anchored-Gemini	5	0.05	0.1	0.5	5	0.05	0.1	0.5	10	0.01	0.05	0.6	5	0.01	0.1	0.6
	Representation-Gemini	20	0.05	0.1	0.5	5	0.05	0.1	0.5	10	0.01	0.05	0.9	5	0.05	0.1	0.65
	Anchored-Hidden	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	Representation-Hidden	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
GPT-4o-mini	Confidence	—	0.01	0.1	0.5	—	0.1	0.1	0.5	—	0.01	0.05	0.5	—	0.05	0.4	0.5
	Anchored-MiniLM	2	0.015	0.1	0.5	2	0.08	0.1	0.7	1	0.05	0.05	0.7	10	0.1	0.25	0.8
	Representation-MiniLM	2	0.01	0.2	0.3	2	0.01	0.2	0.3	2	0.01	0.2	0.3	2	0.01	0.2	0.3
	Anchored-Gemini	5	0.015	0.1	0.5	5	0.005	0.1	0.5	10	0.01	0.05	0.5	20	0.05	0.1	0.8
	Representation-Gemini	20	0.01	0.05	0.5	10	0.01	0.05	0.5	10	0.01	0.05	0.5	5	0.05	0.1	0.8
	Anchored-Hidden	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	Representation-Hidden	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 3: Hyperparameters across all base agents and tasks. γ_t is the calibration-on-the-fly step size, η_t the threshold step size, μ the exploration probability, λ_0 the initial threshold. Confidence has no learnable parameters; its η_t cell is left blank.

C.8 Task implementation details and example prompts

We instantiate each of the four tasks from Section 5. The base agent producing y_0 and y_1 is one of Qwen-2.5-7B, Gemini-2.5-Flash, or GPT-4o-mini, identical across all four tasks. The support modality varies by task: revealed follow-up questions, examination findings, and laboratory results on DDXPlus; SQL query formulation and execution against the table on WikiSQL; scene-specific object locations and constraints on VirtualHome; and targeted guidance from a stronger reasoner on MATH. For MATH, we use DeepSeek as the stronger reasoner that responds to the base agent's questions; the base agent itself remains one of the three models above and is responsible for both the initial solution y_0 and the revised solution y_1 that incorporates the guidance.

For concreteness, we provide the full set of prompts used in the MATH pipeline below. They cover different stages of the interaction: producing the initial solution y_0, eliciting the self-reported confidence $\hat{g}_{\mathrm{LLM}}$ used as the black-box anchor signal in Section 5.1, deciding whether to seek support (used by the LLM-DECIDES baseline), formulating targeted questions for the stronger reasoner, generating the expert guidance, and producing the revised solution y_1. The prompts for the other three tasks follow the same overall structure, adapted to the task-specific support modality, and will be released alongside the code upon acceptance.

Weak Solver — Initial Solution y_0 and Self-Reported Confidence $\hat{g}_{\mathrm{LLM}}$ (MATH)

```
System:  You are a math competition solver and calibration assistant.

User:  Solve the following problem and assess your need for expert support.

Problem:  {{problem}}

Respond in the following JSON format only, with no additional text:

{
"solution":  "<step-by-step reasoning, with the final answer in \boxed{}>",
"confidence":  <number between 0.0 and 1.0>
}

For the confidence field:  if you could ask a math expert specific questions
about this problem, how likely is it that their response would lead you to a
different, correct answer?  Consider whether you were guessing, whether any
step is unsure, and whether knowing the right technique would change your
answer.  Higher values mean expert input is more likely to change your answer.
```

Weak Solver — Decision to Seek Support (MATH)

```
System:  You are a careful math student.  If there is any doubt about your
work, it is worth asking.

User:  Problem:  {{problem}}
Your work:
{{y0_raw}}
Your answer:  {{y0_answer}}

You can ask a math expert questions about this problem.  Their guidance could
help you find errors or discover a better approach.  Asking costs time but can
prevent submitting a wrong answer.  Thus you should ask if you think it will
improve your final answer, and you should proceed with the current answer if
you think it is unlikely that expert guidance will change and improve your
answer.

Consider:  Are you certain your approach is correct, or were there steps
where you guessed or felt unsure?

Respond with ONLY one of:
REQUEST -- if any step felt uncertain
PROCEED -- if you are confident in every step
```

Weak Solver — Question Formulation (MATH)

```
System:  You are a math student asking your tutor for help.  Ask specific,
targeted questions about parts you're stuck on.  Do NOT ask for the full
solution or the final answer.

User:  You attempted this problem and got {{y0_answer}}:

{{problem}}

Your work:
{{y0_raw}}

You can ask a math expert up to 3 questions about this problem.  They will
help you without solving the entire problem for you.

Ask about the specific steps or concepts you're most unsure about.  For
example:
- ''I used [method] -- is this the right approach?''
- ''I got [result] at this step -- is this correct?''
- ''How should I handle [specific part]?''
- ''What identity or theorem applies to [this expression]?''

Write your questions:
```

Strong Expert — Targeted Guidance (MATH)

```
System:  You are a math tutor.  A student is working on a problem and has
asked you specific questions.  Be maximally helpful -- give clear, direct, and
detailed answers to each of their questions.  If they made an error, point
it out and explain why it's wrong.  If they're using the wrong approach,
tell them the right one and explain the key first step.  If they ask about
a technique, explain it concretely with enough detail that they can apply it.

The ONE thing you must NOT do is solve the entire problem for them or state
the final answer.  Help them get unstuck, but let them finish the last steps
on their own.

User:  Problem:
{{problem}}

Student's questions:
{{student_questions}}
```

Weak Solver — Revised Solution y_1 with Guidance (MATH)

```
System:  You are a math competition solver.  You previously attempted this
problem and got stuck on a specific step, so you asked an expert for guidance.
Now solve the problem from scratch using their advice.  Do NOT repeat your
previous mistakes.  Put your final answer in \boxed{}.

User:  Problem:
{{problem}}

The question you asked the expert:
{{expert_question}}

Expert guidance:
{{expert_answer}}

Solve the problem using this guidance.  Put your final answer in \boxed{}.
```

C.9 Comparison of models of varying strength

In this section, we extend our experiments to GPT-5.6-Luna on DDXPlus. We first examine whether our method maintains control of the missed-support error while reducing support usage relative to LLM-Decides. We then compare GPT-5.6-Luna with GPT-4o-mini to investigate how model capability affects the value of seeking support.

Figure 10 reports the cumulative support rate and missed-support error for GPT-5.6-Luna on DDXPlus. Consistent with the results in the main paper, all score variants approach the target missed-support error level α, while the parameterized score families substantially reduce support usage relative to LLM-Decides. These results demonstrate that the benefits of online calibration persist with a more capable base model.

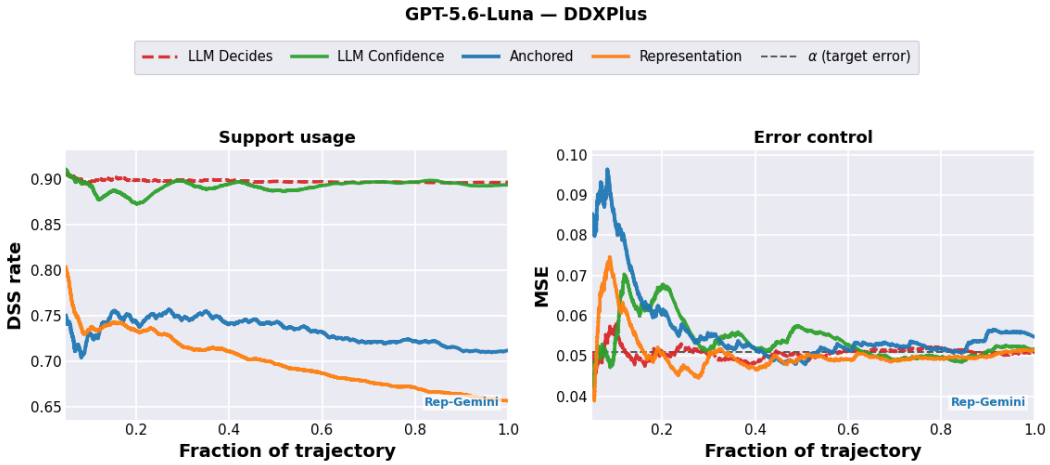

Figure 10: **Error control and support efficiency with GPT-5.6-Luna on DDXPlus.**

As discussed in the main body, the value of support depends jointly on the information provided and the agent's ability to utilize it. To investigate this relationship, we compare GPT-5.6-Luna and GPT-4o-mini on DDXPlus, providing both models with identical support information.

Figure 11 compares the resulting accuracy as a function of support usage. Although GPT-5.6-Luna achieves higher accuracy without support, it also benefits more effectively from the additional information. At comparable support rates, a larger fraction of supported instances lead to improvements for GPT-5.6-Luna than for GPT-4o-mini. Thus, greater model capability does not diminish the

value of support. A stronger model may require less assistance to solve a task independently while simultaneously being better equipped to utilize assistance when it is provided.

These findings further motivate strategic support-seeking policies. As models become more capable, determining *when* support is beneficial remains important, since the value of a support call depends not only on the agent's initial capability but also on its ability to leverage the additional information.

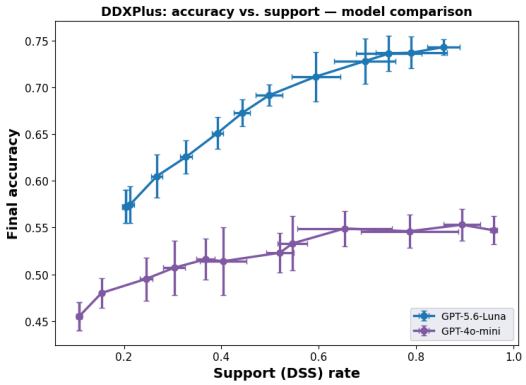

Figure 11: **Effect of model capability on the value of support on DDXPlus.** Accuracy as a function of support usage for GPT-5.6-Luna and GPT-4o-mini under identical support information.

C.10 Experiments with Human Support on ActiveVision

We further evaluate our framework in a human–AI collaboration setting, where support is provided by human participants. We consider the ActiveVision[91] dataset, which consists of challenging visual reasoning puzzles on which even highly capable multimodal language models perform substantially below human-level accuracy. We pair GPT-5.6-Luna with human participants who provide step-by-step guidance in natural language to help the model solve each puzzle.

We collect 118 trials through human crowdsourcing, recording the model's responses both with and without human guidance. This allows us to evaluate the counterfactual benefit of human support and compare different support-seeking policies on the same set of interactions.

Figure 12 reports the cumulative support rate and missed-support error across the interaction trajectory. Consistent with our findings on the other benchmarks, human support meaningfully improves the model's performance, and our algorithm with the best-performing score again outperforms LLM-Decides at matched error.

These results emphasize that our framework can effectively allocate human assistance even when the underlying agent is a highly capable models. In particular, learning when human guidance is beneficial can substantially reduce the number of human interventions required.

We next examine how varying the target missed-support error α affects the resulting accuracy and support usage. Figure 13 reports the final accuracy and support rate across different target error levels. We repeat each experiment over five random permutations of the same 118 trials, reporting the mean and minimum–maximum range across runs.

As expected, stricter error targets lead to increased support usage and higher accuracy. However, the different policies achieve substantially different support rates for comparable accuracy. In particular, at the missed-support error achieved by LLM-Decides, the Anchored policy achieves comparable accuracy while reducing the support rate from approximately 40% to 20%.

C.11 End-to-End Task Performance and the Accuracy–Support Tradeoff

In the main experiments, we evaluate our framework primarily through its ability to control the missed-support error while minimizing support usage. Here, we further investigate how this objective translates into downstream task performance. Specifically, we evaluate the final task accuracy and support rate as we vary the target missed-support error α. This allows us to characterize the tradeoff

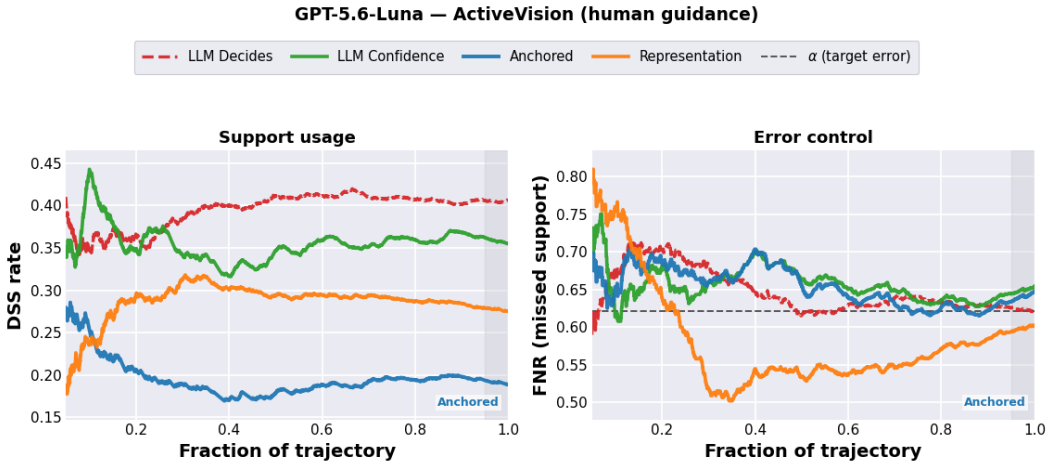

Figure 12: Support efficiency and missed-support error on ActiveVision with human guidance.

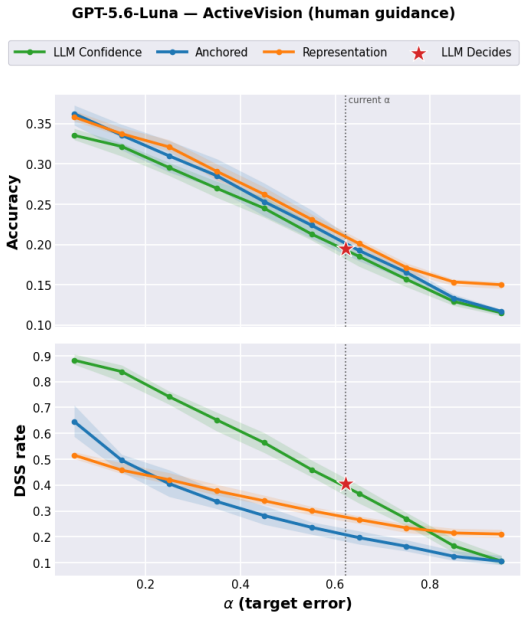

Figure 13: Accuracy–support tradeoff on ActiveVision with human guidance.

between task performance and the cost of seeking support, and to examine how our framework performs under different support budgets.

Figure 14 reports the results for Gemini-2.5-Flash across all four tasks: DDXPlus, WikiSQL, VirtualHome, and MATH. For each task, we use the best-performing parameterized score variant and sweep α over a range of target error levels. We report the final task accuracy (top row) and support rate (bottom row). The red star indicates the performance of LLM-Decides at its empirically achieved missed-support error, providing a reference point for comparing the policies at the same error level.

Across all four tasks, decreasing the target missed-support error generally improves final task accuracy, at the cost of more frequent support calls. Conversely, allowing a higher missed-support error reduces support usage but can degrade downstream performance. This demonstrates that controlling the missed-support error provides a meaningful mechanism for regulating the tradeoff between task accuracy and support cost. Thus, our framework enables practitioners to select operating points that balance downstream performance against the cost or availability of support.

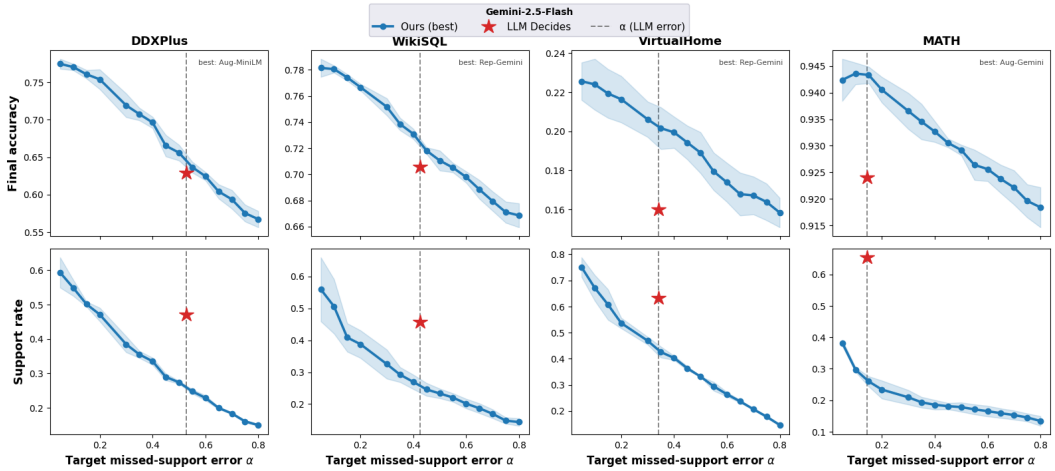

Figure 14: End-to-end accuracy and support usage across target missed-support error levels.

C.12 Operational Variants for the Score Input

The support decision can be made on the basis of any information available at decision time. We outline three natural choices, all of which fit within the framework of Section 3 and inherit the guarantee of Theorem 4.1.

Score on x alone. The score is computed from the input directly, $s = s_\theta(\phi(x))$, before any output is generated. This is the cheapest option, and is appropriate when y_0 is expensive to produce relative to the score itself, e.g., when it requires a long reasoning chain or an expensive tool call.

Score on (x, y_0). The agent generates y_0 first and then scores (x, y_0). This costs an extra forward pass per round but gives the score strictly more information.

Score on $(x, \tilde{y}_0)$ for a cheap surrogate $\tilde{y}_0$. When y_0 itself is expensive, the score can use a cheap surrogate available at decision time, such as the first few tokens of a generation, an intermediate reasoning trace, or a smaller model's prediction. Two conditions must hold: $\tilde{y}_0$ must be available before the support decision, and the feedback signal g_t must still be computable on rounds where support is sought.

The threshold-update guarantee in Theorem 4.1 holds for any of these choices. The choice is purely operational as it sets the cost of running the algorithm and the quality of the score, both of which determine how often support must be invoked to hit a given missed-support error.

Figure 15 reports the empirical comparison on Gemini-2.5-Flash, DDXPlus, with the Anchored-Gemini score. We include the three variants above as well as two reference points that drop the input from the score: scoring on y_0 alone, and scoring on the reasoning trace alone. All five variants control the missed-support error at the target α, consistent with Theorem 4.1. On support efficiency, the three variants that include x or the reasoning trace (with or without y_0) achieve essentially identical support rates, while the variants that is only based on generated y_0 is noticeably worse. The takeaway is that x or the reasoning traces carry most of the signal needed to predict whether support would be beneficial: adding y_0 or the reasoning trace on top of it does not help, but removing x does hurt. From a practical standpoint, this is a useful finding, since scoring on x alone is the cheapest of the variants that retain efficiency: the support decision can be made before any generation is committed. This finding is however task specific- and in practice this might change depending on the underlying task.

Figure 15 reports the empirical comparison on Gemini-2.5-Flash, DDXPlus, with the Anchored-Gemini score. We include the three variants above as well as two reference points that drop the input from the score: scoring on y_0 alone, and scoring on the reasoning trace alone. All five variants control the missed-support error at the target α, consistent with Theorem 4.1. On support efficiency, the variants based on x or the reasoning trace (with or without y_0) achieve essentially identical support rates, while the variant based on y_0 alone is noticeably worse. The takeaway is that x and the reasoning trace each carry most of the signal needed to predict whether support would be beneficial, while y_0 alone does not. From a practical standpoint, this is a useful finding, since scoring on x alone is the cheapest of the efficient variants: the support decision can be made before any generation is committed. This finding is, however, task-specific, and the relative informativeness of x, y_0, and the reasoning trace may differ depending on the complexity of the underlying task in practice.

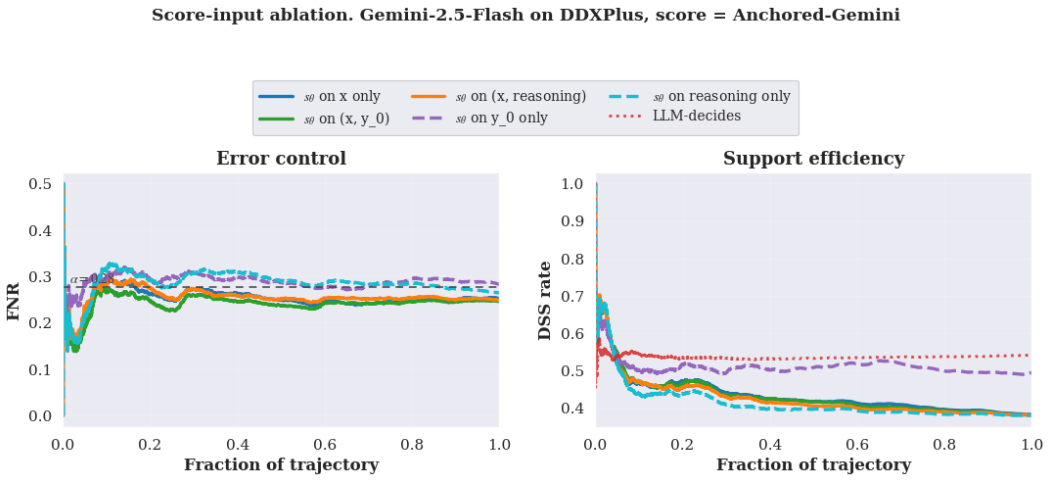

Figure 15: Score-input ablation. Base agent: Gemini-2.5-Flash. Task: DDXPlus. Score: Anchored-Gemini. Left: cumulative missed-support error against the target α. Right: cumulative support rate, with the LLM-DECIDES baseline shown for reference. The five variants score the input alone (s_θ on x), the input together with the unsupported outcome (s_θ on (x, y_0)), the input together with the agent's reasoning trace (s_θ on $(x, \text{reasoning})$), the unsupported outcome alone (s_θ on y_0), or the reasoning trace alone (s_θ on reasoning). All five variants control the missed-support error at α. Variants that include x achieve similar support rates; dropping x degrades efficiency.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All of our contributions and claims are properly explained in the abstract and introduction

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our current work and possible avenues for future research in a separate section called Limitations and Future Work

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Due to space limit, we have moved all the proofs to the supplementary material, where we show full derivations of the proofs carefully, and clearly state all assumptions. We provide references for technical details as needed.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Our experimental setting is well explained in the main body of the paper, with specifics and all extra details of the implementations provided in the Appendix due to space limit. Further, we will publish our code for the camera ready version in case of acceptance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will publish our code for the camera ready version in case of acceptance, and we will ensure instructions are clearly stated.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All details needed for a self-contained experiments section are provided in the main body, while further specifics and implementation details including all hyperparameters are provided in the Appendix for completeness and exact reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes we provide all necessary applicable statistics for a fair comparison.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: To generate responses from the LLM, we can rely on API access which would remove the need for local GPU resources. The estimated total run-time of the experiments for this paper is in order of 3-5 hours, with majority of the time spent querying the LLM. The algorithm itself after these generations, runs in a matter of few minutes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper has no forseeable ethical issues.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Our framework is motivated by a societal concern which is ensuring that AI agents acting on behalf of users remain reliable and aligned with human intent as autonomy increases which will have positive societal impacts as discussed in Section 1. We do not anticipate any negative societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We do not see any foreseeable need for safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited and credited all models and datasets used in our experiments.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: There is no new asset as an output.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: our experiments do not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not include such studies.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods do not involve LLM as any important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.