E-valuator: Reliable Agent Verifiers with Sequential Hypothesis Testing

Anonymous Authors[1]

Abstract

Agentic AI systems execute a sequence of actions, such as reasoning, coding or tool calls, in response to user prompts. To evaluate the success of their trajectories, researchers have developed verifiers, such as process-reward models, to score the quality of each action in an agent's trajectory. Although these heuristic scores can be informative, there are no guarantees of correctness when used to decide whether an agent will yield a successful output. Here, we introduce *e-valuator*, a method to convert any black-box verifier score into a decision rule with provable control of false alarm rates. We frame the problem of distinguishing a successful trajectory—that is, a sequence of actions that will lead to a correct response to the user's prompt—from an unsuccessful trajectory as a sequential hypothesis testing problem. *E-valuator* develops a sequential hypothesis test that remains valid at every step of an agent's trajectory, enabling online monitoring of agents. Empirically, we demonstrate that *e-valuator* provides greater statistical power and better false alarm rate control than other strategies across six datasets and three agents. We additionally show that *e-valuator* can quickly terminate unsuccessful trajectories to save tokens. Together, *e-valuator* provides a lightweight framework that converts verifier heuristics into decision rules with statistical guarantees, enabling the deployment of reliable agents.

1. Introduction

Agents are black-box systems that autonomously perform tasks by executing a sequence of actions, called a trajectory. The actions may include interacting with an external environment through tool calls, writing code, or steps of logical reasoning. In this work, we use *agents* to specifically denote large language model (LLM)-based systems that respond to user requests; more broadly, the *agent* can refer to robotic agents that execute physical tasks (e.g., picking up a cup) through sequences of mechanical actions (Bekey, 1998) and game-playing agents that play games (e.g., chess) through sequences of game-legal actions (Brown & Sandholm, 2019; Silver et al., 2017). Agents have wide applications, and have delivered promising early results across diverse domains, including identifying drug candidates to target acute myeloid leukemia and other diseases (Gottweis et al., 2025; Swanson et al., 2025), mining spatial transcriptomic datasets (Wang et al., 2025; Huang et al., 2025b), and verifying scientific hypothesis (Huang et al., 2025a).

Nonetheless, agents make mistakes, and it is important to be able to detect these. To this end, *verifier* models have been developed to numerically score each action in an agent's trajectory. These scores are typically used as a proxy for the probability that the trajectory will successfully produce a correct final output. Example verifiers include judge LLMs, which provide a score (as text output) after each step (Li et al., 2025), and process-reward models, which are fine-tuned to give a prediction of whether each step in a trajectory is "correct" or "incorrect" (Li & Li, 2025; Zheng et al., 2024; Lightman et al., 2023). These verifiers' scores can then be used to identify unsuccessful trajectories, as those that will produce an incorrect final output.

A key limitation of verifiers to date is that when their scores are used to decide whether to flag a trajectory as incorrect, there are no guarantees on the resulting probability of error of this downstream decision. Rigorous guarantees may become particularly critical when agents are deployed in high-stakes settings with real-world implications, such as autonomous labs (Leong et al., 2025), gene editing (Qu et al., 2025), or hospital operations (Gebreab et al., 2024). In particular, we focus on guaranteeing control over the *false alarm rate*, or the probability of incorrectly flagging a successful trajectory as unsuccessful, as a way of ensuring that any alarms are statistically trustworthy.

Even if verifier scores satisfy popular notions of probabilistic "correctness," such as marginal calibration (**?**), such notions do not provide guarantees on the false alarm rate. Furthermore, each action in a trajectory costs time and re-

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

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

sources, and trajectories can be long. We therefore want to detect that a trajectory will be unsuccessful early on—ideally, after as few actions as possible—instead of having to incur the costs of rolling out the full trajectory before making the decision. These desiderata necessitate assessing the verifier score after each action, a continual sequential testing problem that exacerbates the false alarm issues of naive approaches, particularly since the length of the complete trajectory is not known in advance.

Fine-tuning or building better verifiers does not directly provide error guarantees. Furthermore, fine-tuning verifiers requires sufficient compute as well as white-box access to the verifier (and possibly the agent) weights. Even with sufficient compute and access to the verifier weights, it may be impractical to obtain sufficient data appropriate for fine-tuning verifiers: one needs a "correctness" label for every action in every trajectory (Lightman et al., 2023).

To tackle these challenges, we introduce *e-valuator*, a lightweight statistical wrapper that converts scores from any black-box verifier into a decision rule for detecting unsuccessful agent trajectories, with guarantees on the false alarm rate. To do so, we first frame the problem as a hypothesis test. We assume that the sequence of verifier scores is drawn from one distribution, p_1, for successful trajectories (i.e., those that will produce a correct final output), and another distribution, p_0, for unsuccessful trajectories. Given a new trajectory, the problem then reduces to deciding, after as few actions as possible, whether the verifier score sequence is drawn from p_1 or p_0.

In designing this hypothesis test, we make two contributions. First, we provide a probably-approximately-correct (PAC) thresholding procedure that controls the false alarm rate with high probability for *any* sequential scoring process, using a small calibration set. Second, we draw on prior work in sequential hypothesis testing and e-processes (Ramdas & Wang, 2024; Wald & Wolfowitz, 1948) to construct a specific scoring process—the density ratios between p_1 and p_0 for the partial trajectories at each step—that is *log-optimal*, meaning it accumulates evidence against unsuccessful trajectories as fast as possible. In practice, p_1 and p_0 are not known, so we learn a model of this ratio from the calibration set and account for potential estimation error via our PAC thresholding procedure.

E-valuator builds on approaches from e-values and sequential hypothesis testing. E-values are an alternative to p-values that are particularly useful in settings where one wants to run a sequence of hypothesis tests (e.g., "is this ongoing trajectory successful?") but might not know the number of tests beforehand or may want a cummulative metric of evidence over time. In our setting, for instance, trajectories are of variable length, and a practitioner may wish to either terminate unpromising trajectories as soon as possible or continually track a metric of reliability.

Importantly, *e-valuator* complements any ongoing and future improvements to verifiers: while it guarantees control of the false alarm rate for any combination of agent and verifier, deploying it with better verifiers will tend to yield decision rules with higher power—that is, higher rates of correctly flag unsuccessful trajectories as unsuccessful. *E-valuator* requires minimal compute and can run on a standard laptop.

Empirically, across six datasets and three agents, *e-valuator* provided better false alarm rate control and statistical power than other baselines, such as a raw verifier or re-calibrated verifier model alone. We highlight one additional use case of *e-valuator* with early termination of unsuccessful trajectories, which enables recovering up to 90% of the model's original accuracy with just 80% of the tokens.

2. Methods

2.1. Problem Setting

Given a user prompt, denoted o_0, the agent executes a sequence of $T \in \mathbb{N}_+$ actions, $(a_1, \ldots, a_T)$, where $\mathbb{N}_+ := \{1, 2, ...\}$ and T is a random variable depending on both the prompt and the agent's internal randomness. Associated with each action, a_t, is an observation, o_t, which captures the environment state after the action is executed (e.g., if a_t performs some intermediary arithmetic calculation, o_t could contain the calculated value). The actions and observations together form the *trajectory* at each step t, $H_t = (o_0, a_1, \ldots, a_t, o_t)$. After each step t, a black-box verifier model v takes as input the trajectory H_t and provides a score, $S_t = v(H_t)$, which serves as a heuristic evaluation of the quality of the trajectory thus far. Typically, $S_t \in [0, 1]$, although *e-valuator* supports score values of any type. The verifier scores form a sequence $\mathbf{S} = (S_1, \ldots, S_T)$, and we denote $\mathbf{S}_{[1:t]} = (S_1, \ldots, S_t)$. A complete trajectory H_T is associated with a binary label, $Y \in \{0, 1\}$, of whether the final output, o_T, is correct ($Y = 1$) or not ($Y = 0$). We call trajectories where $Y = 1$ *successful* trajectories and those with $Y = 0$ *unsuccessful* trajectories.

2.2. Evaluation via hypothesis testing

Given a new score sequence, $\mathbf{S}$, our goal is to determine whether the agent's trajectory will produce a correct ($Y = 1$) or incorrect ($Y = 0$) final output. We formalize this goal as a hypothesis test. Let P_1 and P_0 denote the distributions of score sequences conditioned on correct and incorrect final outputs, respectively, with respective densities p_1 and p_0. That is, $p_1(\mathbf{S}) = p(\mathbf{S} \mid Y = 1)$ and $p_0(\mathbf{S}) = p(\mathbf{S} \mid Y = 0)$.

Assuming $P_1 \neq P_0$[1], we test between a "null" hypothesis,

[1] If $P_1 = P_0$, the score sequences would be identically distributed regardless of whether the trajectory is successful or not,

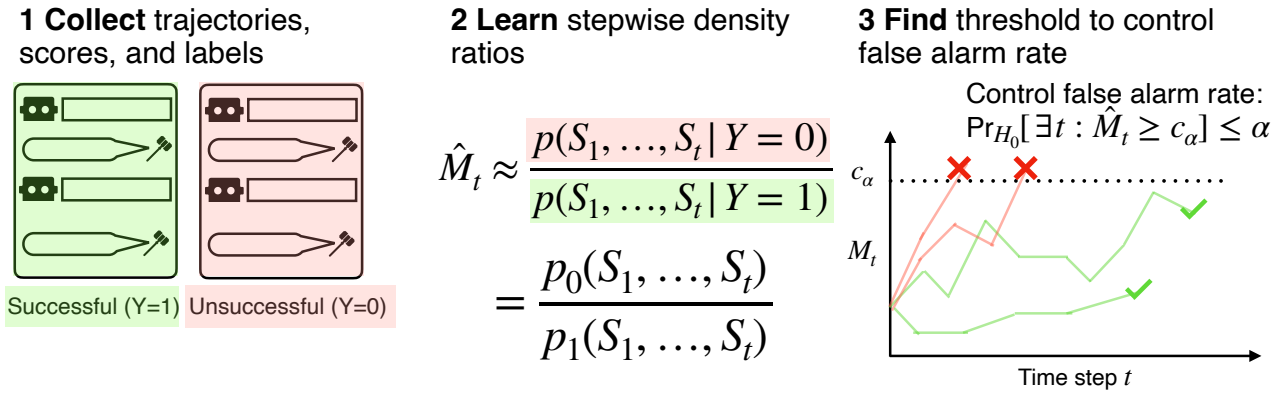

Figure 1. **Overview of e-valuator**. *E-valuator* works in three steps. First, we collect a small calibration set of trajectories, verifier scores, and labels. Second, we learn the density ratios $\hat{M}_t$ at each timestep t. Third, we find a decision threshold that controls the false alarm rate with high probability. Using this threshold, unsuccessful trajectories (red) are rejected at a higher rate than successful ones (green).

$\mathcal{H}_N$, and an "alternative" hypothesis, $\mathcal{H}_A$:

$$\mathcal{H}_N : \mathbf{S} \sim P_1 \quad \text{(the final output is correct)}$$
$$\mathcal{H}_A : \mathbf{S} \sim P_0 \quad \text{(the final output is incorrect).}$$

Note that P_1 and P_0 generally encode complex dependencies between the scores over time. That is, black-box verifier scores are generally not independent samples from a fixed distribution at every step, or otherwise amenable to convenient assumptions.

We construct a *sequential test* between $\mathcal{H}_N$ and $\mathcal{H}_A$ that can be run at each step t, using only $\mathbf{S}_{[1:t]}$, the scores based only on the agent's trajectory up to step t. Specifically, we construct a sequence of test statistics, $(M_t)_{t=1}^T$ where M_t is the output of a real-valued function of $\mathbf{S}_{[1:t]}$, and a real-valued decision threshold, c_α, given a user-specified error level, $\alpha \in (0, 1)$. For $t = 1, 2, \ldots$, if $M_t \geq c_\alpha$, we reject $\mathcal{H}_N$. If we reach $t = T$, the end of the agent's trajectory, without rejecting $\mathcal{H}_N$, then we accept $\mathcal{H}_N$. Our primary goal in designing this sequence $(M_t)_{t=1}^T$ and selecting the threshold c_α is to control the **false alarm rate**. We define **false alarm rate (FAR) control** as

$$\Pr_{\mathcal{H}_N}[\exists\, t \in [T] : M_t \geq c_\alpha] \leq \alpha \tag{1}$$

where T, the length of the complete trajectory, is also random, and $[T] := \{1, \ldots, T\}$. In words, the *false alarm rate*, or the rate at which we reject successful trajectories, is guaranteed to be at most α. Notice that the probability under consideration is that M_t *ever* surpasses c_α—equivalently, that we *ever* reject the null hypothesis, $\mathcal{H}_N$—regardless of T, the total number of actions the agent takes (which varies from trajectory to trajectory).

meaning the verifier scores carry no information about correctness. In this case, no test based on the scores can do better than random guessing.

Of course, controlling the false alarm rate alone is insufficient; the trivial decision rule that never rejects has a false alarm rate of zero. Indeed, the choice of $(M_t)_{t=1}^T$ determines how quickly the test accumulates evidence against the null ($\mathcal{H}_N$) when the trajectory is unsuccessful ($\mathcal{H}_A$). We will return to the choice of $(M_t)_{t=1}^T$ in Section 2.4.

Concretely, our method randomly splits our calibration data, $\mathcal{D}_{\text{cal}}$, into disjoint sets, $\mathcal{D}_{\text{DRE}}$ and $\mathcal{D}_{\text{threshold}}$, such that $\mathcal{D}_{\text{cal}} = \mathcal{D}_{\text{DRE}} \cup \mathcal{D}_{\text{threshold}}$. We design the scoring process, $(M_t)_{t=1}^T$, based on $\mathcal{D}_{\text{DRE}}$ (Section 2.4), and select the decision threshold, c_α, using $\mathcal{D}_{\text{threshold}}$ (Section 2.3).

2.3. False alarm rate control via PAC thresholding

In this section, we provide a procedure that controls the FAR (Eq. 1) for *any* sequential process $(M_t)_{t=1}^T$. In particular, let $(f_t)_{t=1}^T$ be a sequence of functions where $f_t : \mathbb{R}^t \to \mathbb{R}$ takes as input the (partial) sequence $\mathbf{S}_{[1:t]} \in \mathbb{R}^t$ and returns $M_t \in \mathbb{R}$. For instance, f_t could be the identity function, such that we consider the raw verifier scores, $M_t = f_t(\mathbf{S}_{[1:t]}) = S_t$. We will return to the design of $(f_t)_{t=1}^T$ in Section 2.4.

Our goal is to find a threshold c_α for any $\alpha \in (0, 1)$, such that $\Pr_{\mathcal{H}_N}[\exists\, t \in [T] : M_t \geq c_\alpha] \leq \alpha$. Note that rejecting $\mathcal{H}_N$ if the process M_t ever surpasses c_α is equivalent to rejecting $\mathcal{H}_N$ if $\max_t M_t \geq c_\alpha$. Thus, it suffices to set c_α to the $(1 - \alpha)$ quantile of the distribution of $\max_t M_t$ under the null.

Accordingly, to set c_α, we focus on successful trajectories corresponding to $\mathcal{H}_N$. For each ith trajectory in the threshold calibration set $\mathcal{D}_{\text{threshold}}$, we record the maximum

Algorithm 1 Probably-approximately-correct (PAC) threshold.

Inputs: error level $\delta \in [0,1]$; quantile level $\alpha \in [0,1]$; calibration data $\mathcal{D}_{\text{threshold}}$; functions that score the process at each step t, $\{f_t\}$.
Output: decision threshold c_α that achieves FAR control with high probability.
$i \leftarrow 0$
for $(\mathbf{S}^{(i)}, Y^{(i)}) \in \mathcal{D}_{\text{threshold}}$ such that $Y^{(i)} = 1$ **do**
  $M_1, \ldots, M_T \leftarrow f_1(\mathbf{S}^{(i)}_{[1:1]}), \ldots, f_T(\mathbf{S}^{(i)}_{[1:T]})$
  $M^{(i)} \leftarrow \max_t M_t$ ▷ Maximum for i-th null sample
end for
Sort $M^{(1)}, \ldots, M^{(n)}$ in ascending order, $M_{(1)} \leq \cdots \leq M_{(n)}$. Break ties by flipping a fair coin.
$k \leftarrow \min\{j \in [n] : \Pr[\text{Bin}(n, 1-\alpha) \geq j] \leq \delta\}$ ▷ Find index using Binomial tail bound.
$c_\alpha \leftarrow M_{(k)}$

M_t over all steps, $M^{(i)} = \max\{M_1^{(i)}, \ldots, M_T^{(i)}\}$. This maximum value, $M^{(i)}$, is a sample from the null distribution. Given these samples, we construct $\hat{q}_{1-\alpha}$, a high-probability upper bound on the $(1-\alpha)$-quantile of the null distribution (Alg. 1). We refer to this procedure as **probably-approximately-correct (PAC)** thresholding. In particular, with probability at least $1-\delta$ for any user-specified δ (that is, "probably"), the procedure has anytime control of the false alarm rate under a user-specified α ("approximately correct").

Proposition 2.1. *Let $\{M_t\}_{t\in\mathbb{N}_+}$ denote any scoring process where $M_t = f_t(\mathbf{S}_{[1:t]})$ for a sequence of (deterministic) functions $(f_t)_{t\in\mathbb{N}_+}$ and $M_t \in \mathbb{R}$. For fixed error level $\delta \in (0,1)$ and quantile level $\alpha \in (0,1)$, let c_α be the output of Algorithm 1. Then,*

$$Pr_{\mathcal{D}_{cal}}\left(Pr_{\mathcal{H}_N}\left(\exists\, t \in [T] : M_t \geq c_\alpha \mid \mathcal{D}_{cal}\right) \leq \alpha\right) \geq 1-\delta.$$

See Appendix A.1.1 for the proof. Observe that Proposition 2.1 involves two error parameters: α (the quantile level) and δ (the probability of FAR miscalibration). By a union bound, the marginal false alarm rate satisfies $\Pr_{\mathcal{H}_N}[\exists\, t \in [T] : M_t \geq c_\alpha] \leq \alpha + \delta$. Thus, to control the marginal false alarm rate at a desired level α_0, one can choose any α and δ such that $\alpha + \delta \leq \alpha_0$.

2.4. Designing a (log-)optimal test statistic

Although Proposition 2.1 controls the false alarm rate with high probability for any functions $(f_t)_{t\in\mathbb{N}_+}$, it does not guide us in *which* functions we should use. Recall that our test rejects a trajectory when M_t first exceeds c_α. Thus, we wish to design $(f_t)_{t\in\mathbb{N}_+}$ such that under $\mathcal{H}_A$—that is, when the trajectory is unsuccessful—M_t grows rapidly.

To do so, we draw from *e-processes* (Ramdas & Wang, 2024), a framework for sequential hypothesis testing that enables principled construction of $(f_t)_{t\in\mathbb{N}_+}$ and additional theoretical guarantees.

An e-process for $\mathcal{H}_N$ is a non-negative stochastic process, $(E_t)_t$, such that each E_t is an *e-value* for $\mathcal{H}_N$—that is, $\mathbb{E}_{\mathcal{H}_N}[E_t] \leq 1$—and there exists a *test martingale* for $\mathcal{H}_N$, $(M_t)_t$, such that $E_t \leq M_t$ always. A test martingale for $\mathcal{H}_N$ is a sequence, $(M_t)_t$, that satisfies two conditions: (1) **Non-negativity and unit mean**: M_t is non-negative for all t, and $E_{\mathcal{H}_N}[M_0] \leq 1$. (2) **Martingale**. $(M_t)_t$ is a martingale for $\mathcal{H}_N$. For all t, $\mathbb{E}_{\mathcal{H}_N}[M_t \mid M_0, \ldots, M_{t-1}] = M_{t-1}$. [2]

Note that any test martingale is an e-process, but that e-processes are a much broader class of processes.

E-processes are particularly useful as their concentration behavior can be exploited to provide direct FAR control. In particular, when $(M_t)_{t=1}^\infty$ is an exact e-process, Ville's inequality (Ville, 1939) gives a distribution-free and time-uniform threshold: $\Pr_{\mathcal{H}_N}[\exists\, t \in \mathbb{N}_+ : M_t \geq 1/\alpha] \leq \alpha$. That is, setting $c_\alpha = 1/\alpha$ achieves FAR control without any calibration data when $(M_t)_{t=1}^\infty$ is an exact e-process.

The density ratio process provides a natural test martingale for our setting. In particular, we use the ratio between the alternative density, p_0, and null density, p_1 at each step t. Specifically, set $M_0 = 1$, and, for each step $t \in [T]$,

$$M_t = f_t(\mathbf{S}_{[1:t]}) = \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})}. \tag{2}$$

It turns out that under $\mathcal{H}_A$, the density ratio process grows the fastest (in expectation, log-scaled) over time among all e-processes. That is, if $M_t = \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})}$ denotes the density ratio process, then for any other e-process $(M'_t)_{t=0}^T$ and stopping time τ, $E_{\mathcal{H}_A}[\log M_\tau] \geq E_{\mathcal{H}_A}[\log M'_\tau]$. See Appendix A.1.2 for a formal statement and proof of this.

Log-optimality is analogous to being the "most powerful" test statistic in non-sequential hypothesis testing: intuitively, M_t will tend to surpass the decision threshold, and correspondingly enable detection of unsuccessful trajectories earlier than other e-processes.

In practice, the forms of p_1 and p_0 are typically not known. Prior to deployment, the method we propose, *e-valuator*, therefore has a calibration phase in which it uses a split of

[2] More formally, a martingale satisfies $E_{\mathcal{H}_N}[M_t|\mathcal{F}_{t-1}] = M_{t-1}$ for all t, where $(\mathcal{F}_t)_t$ is a filtration. For clarity of exposition and space constraints, we avoid a fully rigorous treatment here, but roughly, this means we are conditioning on all the information available in $\mathcal{F}_{t-1}$, which may be more or less information than conditioning on the exact values of the past. This is equivalent to conditioning on the exact values of the past when $\mathcal{F}_t$ is the so-called natural filtration.

the calibration data, $\mathcal{D}_{\text{DRE}}$, to learn a model of the density ratio, $\hat{M}_t(\mathbf{S}_{[1:t]}) \approx \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]}))}$, for each step t. Indeed, even if one cannot exactly evaluate the true densities, one can learn the density ratios and approximate the log-optimal tests, as is done in prior works (**?**). We expand on the density ratio estimation below.

2.5. Density ratio estimation

To estimate our density ratios, we use classifier-based density ratio estimation. In particular, observe the following equality by Bayes' rule (Bickel et al., 2009; Gutmann & Hyvärinen, 2012):

$$M_t = f_t(\mathbf{S}_{[1:t]}) = \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})} = \frac{p(Y=0|\mathbf{S}_{[1:t]})}{p(Y=1|\mathbf{S}_{[1:t]})} \cdot \frac{p(Y=1)}{p(Y=0)}. \tag{3}$$

Thus, for each time step t, we train a classifier, $\hat{g}_t$, which takes $\mathbf{S}_{[1:t]}$ as input and provides an estimate of $p(Y = 1|\mathbf{S}_{[1:t]})$. We also form an estimate, $\hat{\pi}_1$, of the class probability $p(Y = 1)$ from the calibration set. We plug these two estimates into Eq. (3) to form the following estimated density ratio at step t:

$$M_t = \hat{f}_t(\mathbf{S}_{[1:t]}) = \underbrace{\frac{1-\hat{g}_t(\mathbf{S}_{[1:t]})}{\hat{g}_t(\mathbf{S}_{[1:t]})}}_{\text{classifier-based estimate}} \cdot \underbrace{\frac{\hat{\pi}_1}{1-\hat{\pi}_1}}_{\text{prior odds estimate}}$$

We then select the threshold c_α using the procedure in Proposition 2.1, which applies to *any* scoring process with i.i.d. trajectories from the null available in the calibration set. In our experiments, we used simple logistic regression for the classifier at each step, $\hat{g}_t$, and found that estimated density ratios learned from a few hundred calibration points empirically achieved both false alarm rate control and superior power to alternative methods (see Appendix A.2.3).

3. Related Work

As *e-valuator* is a statistical wrapper for any verifier, our work is relevant for but orthogonal to prior work on building better verifiers for agents. These verifiers are often trained as reward models that estimate a reward (e.g., correctness or coherence) after each step in an agent's action sequence. Among these, process-reward models (PRMs) are finetuned using agent trajectories where each *step* is labeled as correct or incorrect, as in a mathematical reasoning trace (Uesato et al., 2022; Wang et al., 2024a; Khalifa et al., 2025). Some prior works also work to calibrate existing PRMs (You et al., 2025; Park et al., 2025), although calibration alone is insufficient to control false alarm rates, as we do in this paper. Training PRMs can be expensive, as it requires (a) access to human-annotated process labels (Lightman et al., 2023) and (b) finetuning existing LLMs (Wang et al., 2024a). Alternative verifiers to PRMs include LLM-as-a-judge (i.e., prompt-based verification) (Bavaresco et al., 2024) and outcome reward models, which only provide a label for the entire trajectory (Creswell et al., 2022). There are several benchmarks to compare verifiers on trajectories (Lù et al., 2025; Zheng et al., 2024).

Some prior works build hypothesis tests atop classifiers to monitor AI deployments, albeit in different contexts. Vovk et al. (2021) propose using conformal test martingales (CTMs) for continual monitoring of model deployments, and Prinster et al. (2025) develop weighted CTMs to enable adaptation at test time and analyze the cause of degradation. Podkopaev & Ramdas (2021) leverage sequential testing for directly tracking the risk of deployed models, which has been extended to monitoring without labels (I Amoukou et al., 2024), under test-time adaptation (Schirmer et al., 2025), and for unknown shifts (Timans et al., 2025). In a similar vein, (Chugg et al., 2023) applies safe anytime-valid testing to sequentially test if a classifier is fair. (Jang et al., 2022) applies classifier two-sample tests (Lopez-Paz & Oquab, 2016) to detect whether there is a covariate shift between the training and deployment setting. More broadly, several prior works model class-conditional densities of classifier scores and the corresponding density ratios for evaluation of AI systems, albeit without hypothesis testing (Shanmugam et al., 2025; Welinder et al., 2013; Dawid & Skene, 1979; Ji et al., 2020).

Another line of work tries to add formal statistical controls to LLMs. Conformal prediction methods (Shafer & Vovk, 2008) quantify uncertainty in individual predictions while providing finite-sample and black-box guarantees in those uncertainties. Conformal prediction has been applied to resample LLM generations until a minimum quality requirement is satisfied (Quach et al., 2023). These ideas have been further applied to control the factuality of LLM outputs (Cherian et al., 2024; Mohri & Hashimoto, 2024; Ye et al., 2024). More recently, Wu et al. (2025) applies the learn-then-test framework (Angelopoulos et al., 2025) to calibrate a stopping rule for LLM reasoning traces, using white-box access to the LLM's internal logits.

E-valuator directly builds on prior work in e-values, originally introduced as an alternative to p-values, with useful properties for sequential hypothesis testing (Ramdas & Wang, 2024; Vovk & Wang, 2021; 2023). E-values provide anytime validity over a (potentially infinite) sequence of tests (Wang & Ramdas, 2025; Waudby-Smith et al., 2024; 2023; Grünwald et al., 2020). They have found important applications in A/B testing (Johari et al., 2017), changepoint detection (Shin et al., 2023; Lorden, 1971), detection of LLM-generated text (Chen & Wang), LLM-based hypothesis verification (Huang et al., 2025a), and LLM evaluation

(Dhillon et al., 2025).

4. Experiments

To empirically validate *e-valuator*, we compare it to other baseline methods along three axes: (1) **false alarm rate control**: "does e-valuator maintain the false alarm rate below the desired level, α?" (2) **power**: "how often does e-valuator reject unsuccessful trajectories?" and (3) **token savings**: "among unsuccessful trajectories, how quickly does e-valuator reject, and (relatedly) how many tokens do we save via early rejection?" These are common hypothesis testing metrics that matter in practice: a user would like to catch as many unsuccessful trajectories as possible, and catch them early enough to save compute. Log-optimality (Section 2.4) suggests that the density ratio statistic should accumulate evidence against $\mathcal{H}_A$ quickly, which should manifest empirically as both higher power (more total rejections) and shorter run lengths under $\mathcal{H}_A$ (earlier rejections). We conducted our experiments across six datasets and tasks using three different agent-verifier combinations.

Agents and verifiers. We conduct experiments on two tool-calling agents, Aviary (Narayanan et al., 2024) and Octotools (Lu et al., 2025), one step-by-step reasoning model, Claude Sonnet 4. For Aviary and Octotools, we use Claude Haiku 3.5 as the verifier, asking after each tool call for a text-based probability that the trajectory thus far is successful. For the reasoning model, we use a popular pretrained process-reward model (Wang et al., 2024a), which provides a logit-based probability that each step in a reasoning trace is correct.

Datasets. We experiment on six datasets across two domains: (1) **mathematical reasoning** (GSM8k (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and AIME (**?**)), and (2) **question-answering** (HotpotQA (Yang et al., 2018), MedQA (Jin et al., 2021), and MMLU-Pro (Wang et al., 2024b)). We present results from all datasets except GSM8k and AIME in the main section, and provide the GSM8k and AIME results in Appendix A.2. Results for each dataset are from one agent-verifier combination. A full description of the dataset, agent, and verifier combinations is available in Appendix A.3. Our verifiers span a wide range of quality (Appendix table 2).

Baselines We compare against four baselines inspired by sequential hypothesis testing and calibration.

1. The **raw verifier** uses the scores from the verifier without modification. The verifier provides some prediction of $\Pr(Y = 1|H_t)$, the probability that the agent will produce a successful output, given the trajectory H_t thus far. For a user-specified false positive rate α, we reject a trajectory if the score S_t ever drops below α.

2. The **calibrated verifier** uses the same verifier but recalibrates the scores S_t using the calibration set $\mathcal{D}_{cal}$. Specifically, we use isotonic regression to learn a $[0, 1]$-valued transformation of the score, $S' = \hat{f}(S) \in [0, 1]$, that achieves *marginal calibration*: $\Pr(Y = 1|S') = S'$. As with the raw verifier, for a user-specified false positive rate α, we reject a trajectory if the score S_t ever drops below α.

3. The **PAC verifier** is a variant of our method that uses the procedure from Algorithm 1, setting $f_t(\mathbf{S}_{[1:t]}) = 1 - S_t$ to account for the fact that *smaller* S_t from the verifier indicate "worse" trajectories. We then find c_α with Algorithm 1 and use it as the threshold for rejection.

4. **Randomized Ville** utilizes a modern variant of Ville's inequality (Ramdas & Manole, 2026) to set the rejection threshold c_α. Specifically, the threshold is $1/\alpha$ at all steps except the final step, where we threshold at Z/α (with $Z \sim \text{Unif}(0, 1)$), allowing for more rejections at the end of a trajectory. We provide full details in Appendix A.1.4. As with the standard Ville threshold, this procedure guarantees the false alarm rate control only on exact test martingales; when the density ratios are estimated, false alarm rate control may not hold.

We compare these baselines to **e-valuator**, in which we choose the threshold using the procedure in Proposition 2.1. To meet a user-specified false alarm rate α, we set $\alpha' = 0.9 \cdot \alpha$ and $\delta = 0.1 \cdot \alpha$ so that $\alpha' + \delta = \alpha$ (in our main text experiments). We use the same α', δ split for the PAC verifier.

For the results presented in the main section, we use an 80/20 split of test/calibration data: we calibrate our method (and baselines) on 20% of the data and test on 80%. We compare other splits of the calibration set in Appendix A.2.3 and find that a few hundred calibration trajectories is sufficient to achieve good false alarm rate control and power. We visualize the M_t sequences for both successful and unsuccessful trajectories in Appendix A.2. Finally, we also ablate the density ratio estimator in Appendix A.2.4.

4.1. E-valuator provides better false alarm rate control than alternative methods

We begin by analyzing the empirical false alarm rate achieved by *e-valuator* and the competing baselines. Given a particular user-specified α, the false alarm rate, or the rate at which we flag successful ("null") trajectories as unsuccessful ("alternative"), should be no greater than α.

E-valuator empirically controls the false alarm rate across all choices of α and all datasets (Figure 2, top). The raw verifier,

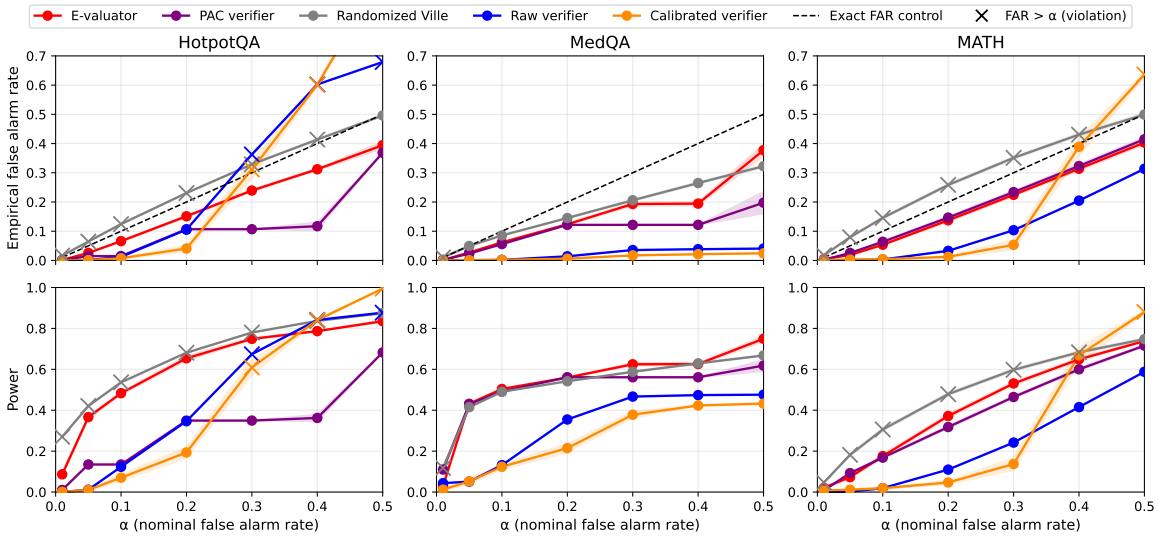

Figure 2. **E-valuator controls the false alarm rate and maximizes power better than alternative methods**. Violations of the false alarm rate control are marked with an X. Randomized Ville, the raw verifier, and the calibrated verifier occassionally provide comparable power, at the cost of inflating the false alarm rate. The PAC verifier successfully controls the false alarm rate but has consistently worse power than e-valuator across the three datasets. All plots show the 95% CI over 50 random splits of each dataset.

which thresholds the raw scores S_t at α, sometimes achieves empirical false alarm rates less than the desired α (MedQA) but not always (HotpotQA, for $\alpha > 0.4$). The calibrated verifier, which applies isotonic regression to the raw verifier scores S_t and then uses the same threshold $c_\alpha = \alpha$ on the recalibrated scores, does not achieve false alarm rates less than α either. Although calibration procedures such as isotonic regression aim to achieve $\mathrm{E}(Y|S) = S$, this property does not have any direct implications on the false alarm rate. Furthermore, even if this property held at each timestep, it does not allow us to reason about the false alarm rate in the sequential hypothesis setting (see Appendix A.1 for further discussion).

Similarly, randomized Ville, which uses the threshold $1/\alpha$ until the final timestep and Z/α for the final timestep (for $Z \sim \mathrm{Unif}(0, 1)$), successfully controls the false alarm rate on one dataset (MedQA) but not others (HotpotQA, MATH). The false alarm rate violations are likely due to density ratio estimation error, as randomized Ville controls the false alarm rate only for exact test martingales, not necessarily approximate ones. The PAC verifier (an application of Algorithm 1 to the raw verifier scores) is indeed able to control the false alarm rate, as expected. However, it suffers in power, as we discuss below.

4.2. E-valuator provides enhanced power over alternate methods

Next we analyze the empirical power across the same datasets and tasks (Figure 2, bottom). That is, power is the rate at which unsuccessful trajectories ("alternative") are indeed flagged as unsuccessful. Across all datasets and all αs, e-valuator provides greatest power among methods that achieve empirical false alarm rates less than α.

In those instances where the competing methods provide better power, it is at the cost of an inflated false alarm rate. For instance, in HotpotQA, at $\alpha = 0.4$, the raw verifier provides a power of 0.84 but inflates the false alarm rate to 0.60. Similarly, randomized Ville has a power of 0.84 at the cost of a false alarm rate of 0.41.

In contrast, *e-valuator* provides a power of 0.78 *and* controls the false alarm rate at 0.31. The PAC verifier successfully controls the false alarm rate (0.12) at the cost of substantially less power (0.36). These findings hold true across datasets. Similar plots comparing false alarm rate control and power are available in Appendix A.2 for the MMLU-Pro, AIME, and GSM8k datasets.

4.3. E-valuator terminates unsuccessful trajectories earlier and recovers a larger fraction of original accuracy in limited token budget

Finally, we evaluate how quickly each method detects unsuccessful trajectories. Whereas power compares the rate at which unsuccessful trajectories are rejected (at some point), in this section we compare how early each method rejects, as measured by the average run length, or average number of steps it takes to reject a trajectory, under $\mathcal{H}_A$. In particular, we terminate any trajectory as soon as $\hat{M}_t$ crosses the threshold c_α. This is practically important: every additional step of an unsuccessful trajectory wastes tokens.

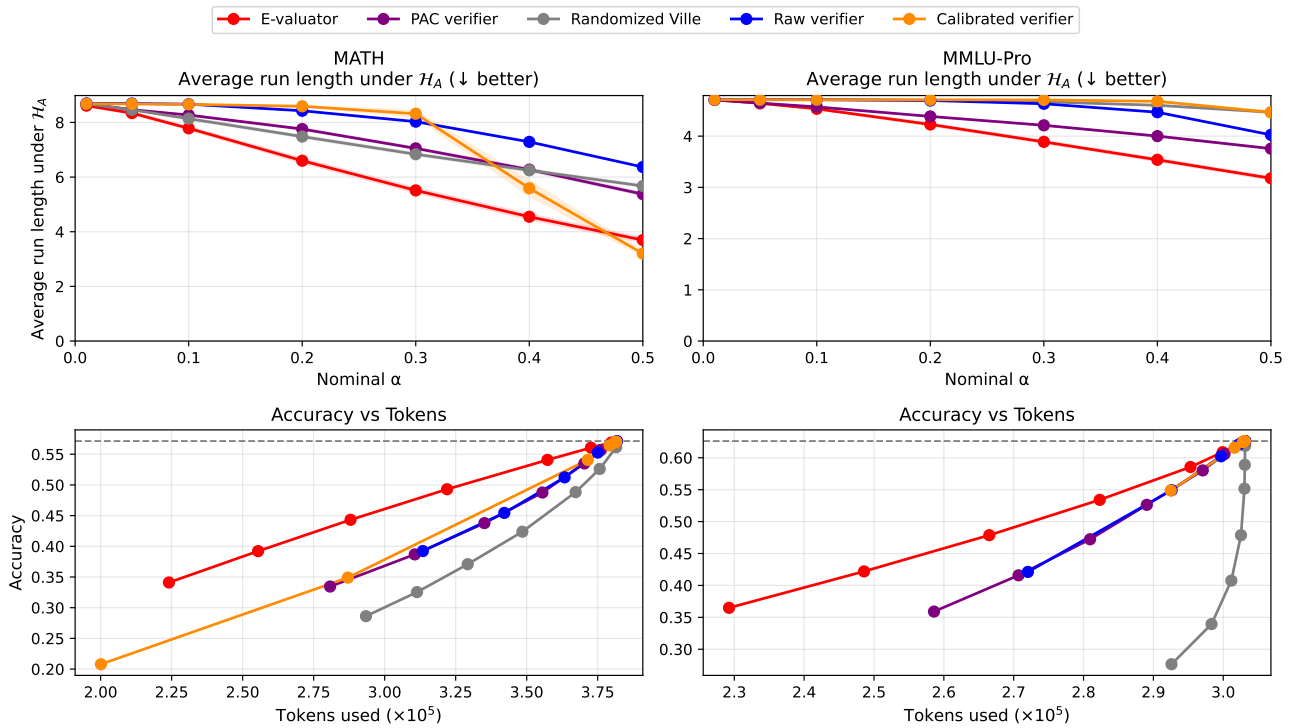

Figure 3. **E-valuator terminates unsuccessful trajectories earlier, providing better token-accuracy tradeoff.** (Top) Average run length under $\mathcal{H}_A$ (↓ better): *e-valuator* rejects unsuccessful trajectories in fewer steps (on average) than competing methods. (Bottom) By terminating unsuccessful trajectories earlier, *e-valuator* recovers a larger fraction of the original accuracy (dashed line) for a given token budget. Results shown on MATH and MMLU-Pro. False alarm rate violations are marked with an X.

E-valuator is able to detect unsuccessful trajectories earlier (Fig 3, row 1). For instance, at $\alpha = 0.3$ on the MATH dataset, *e-valuator* detects unsuccessful trajectories within 5.5 steps on average, whereas all competing baselines reject after at least 6.8 steps on average. Although the calibrated verifier achieves better average run length at $\alpha = 0.5$ on MATH, it fails to control the false alarm rate (Fig. 2).

This results in a better tradeoff between token usage and accuracy (Fig 3, row 2). We count the number of tokens that are saved (i.e., how many tokens would have otherwise been generated if the agent weren't terminated) and compare that to the total accuracy on the dataset. Note that using the full token budget is the "maximal" accuracy achievable, as this would entail never terminating any trajectory.

Comparing *e-valuator* to the verifier in terms of tokens saved versus total accuracy, we find that *e-valuator* outperforms all competing baselines in recovering accuracy with fewer tokens. For instance, on the MATH dataset, *e-valuator* achieved 49% total accuracy (86% of the original accuracy of 57%) using 84% of the original 381,804 tokens. By contrast, the raw, calibrated, and PAC verifiers each require more than 98% of the original token count to recover comparable accuracy. Similarly, on the MMLU-Pro dataset, *e-valuator* achieved 53% of the total accuracy using just 282,316 tokens, whereas the raw, calibrated, and PAC verifiers each require more than 300,000 tokens.

5. Discussion

In this paper, we introduced *e-valuator*, a method to improve any agent verifier model using sequential hypothesis testing. We convert the problem of detecting whether a trajectory is "successful" or not into a hypothesis testing problem, where we distinguish between verifier scores generated from the "successful" versus the "unsuccessful" distribution. We introduce a PAC procedure to control the false alarm rate even with estimated density ratios.

There are promising directions for future work. First, one can relax certain assumptions to avoid estimating the *full* joint density at each time t, such as assuming the verifier scores are i.i.d. across steps, which would allow universal inference algorithms (Wasserman et al., 2020) to construct exact e-processes from estimated density ratios. Second, although we explored a potential test-time scaling application, one could use *e-valuator* for more nuanced strategies, such as resampling bad trajectories. Finally, *e-valuator* can be extended to more complex agentic systems, such as multi-agent settings.

Software and Data

Our results rely on publicly available datasets. We release code and data anonymously in a supplemental zip file.

Impact Statement

We presented a method to verify agent outputs with statistical guarantees, given access to a black-box verifier model and agent. This approach can help control agent behaviors and remove low-quality or incorrect outputs. In addition to helping with agent verification, our work has implications in safety. For instance, one could use our framework to verify the safety of agent outputs (with high probability), given some noisy and black-box safety verification model (e.g., flag unsafe responses, while controlling for false alarm rates).

References

Angelopoulos, A. N., Bates, S., Candès, E. J., Jordan, M. I., and Lei, L. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025.

Bavaresco, A., Bernardi, R., Bertolazzi, L., Elliott, D., Fernández, R., Gatt, A., Ghaleb, E., Giulianelli, M., Hanna, M., Koller, A., et al. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *arXiv preprint arXiv:2406.18403*, 2024.

Bekey, G. A. On autonomous robots. *The Knowledge Engineering Review*, 13(2):143–146, 1998.

Bickel, S., Bruckner, M., and Scheffer, T. Discriminative learning under covariate shift. *J. Mach. Learn. Res.*, 10: 2137–2155, 2009.

Brown, N. and Sandholm, T. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

Chen, C. and Wang, J.-K. Online detection of llm-generated texts via sequential hypothesis testing by betting. In *Forty-second International Conference on Machine Learning*.

Cherian, J., Gibbs, I., and Candes, E. Large language model validity via enhanced conformal prediction methods. *Advances in Neural Information Processing Systems*, 37: 114812–114842, 2024.

Chugg, B., Cortes-Gomez, S., Wilder, B., and Ramdas, A. Auditing fairness by betting. *Advances in Neural Information Processing Systems*, 36:6070–6091, 2023.

Clopper, C. J. and Pearson, E. S. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Dawid, A. P. and Skene, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

Dhillon, G. S., González, J., Pandeva, T., and Curth, A. E-scores for (in) correctness assessment of generative model outputs. *arXiv preprint arXiv:2510.25770*, 2025.

Gebreab, S. A., Salah, K., Jayaraman, R., Habib ur Rehman, M., and Ellaham, S. LLM-based framework for administrative task automation in healthcare. In *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–7. IEEE, April 2024.

Gottweis, J., Weng, W.-H., Daryin, A., Tu, T., Palepu, A., Sirkovic, P., Myaskovsky, A., Weissenberger, F., Rong, K., Tanno, R., et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Grünwald, P., de Heide, R., and Koolen, W. M. Safe testing. In *2020 Information theory and applications workshop (ITA)*, pp. 1–54. IEEE, 2020.

Gutmann, M. U. and Hyvärinen, A. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13 (11):307–361, 2012.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Huang, K., Jin, Y., Li, R., Li, M. Y., Candes, E., and Leskovec, J. Automated hypothesis validation with agentic sequential falsifications. *Forty-second International Conference on Machine Learning*, 2025a.

Huang, K., Zhang, S., Wang, H., Qu, Y., Lu, Y., Roohani, Y., Li, R., Qiu, L., Li, G., Zhang, J., et al. Biomni: A general-purpose biomedical ai agent. *biorxiv*, 2025b.

I Amoukou, S., Bewley, T., Mishra, S., Lecue, F., Magazzeni, D., and Veloso, M. Sequential harmful shift detection without labels. *Advances in Neural Information Processing Systems*, 37:129279–129302, 2024.

Jang, S., Park, S., Lee, I., and Bastani, O. Sequential covariate shift detection using classifier two-sample tests. In *International conference on machine learning*, pp. 9845–9880. PMLR, 2022.

Ji, D., Smyth, P., and Steyvers, M. Can i trust my fairness metric? assessing fairness with unlabeled data and bayesian inference. *Advances in Neural Information Processing Systems*, 33:18600–18612, 2020.

Jin, D., Pan, E., Oufattole, N., Weng, W.-H., Fang, H., and Szolovits, P. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Johari, R., Koomen, P., Pekelis, L., and Walsh, D. Peeking at a/b tests: Why it matters, and what to do about it. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1517–1525, 2017.

Khalifa, M., Agarwal, R., Logeswaran, L., Kim, J., Peng, H., Lee, M., Lee, H., and Wang, L. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Leong, S. X., Griesbach, C. E., Zhang, R., Darvish, K., Zhao, Y., Mandal, A., Zou, Y., Hao, H., Bernales, V., and Aspuru-Guzik, A. Steering towards safe self-driving laboratories. *Nat. Rev. Chem.*, 9(10):707–722, October 2025.

Li, D., Jiang, B., Huang, L., Beigi, A., Zhao, C., Tan, Z., Bhattacharjee, A., Jiang, Y., Chen, C., Wu, T., et al. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, 2025.

Li, W. and Li, Y. Process reward model with q-value rankings. *The Thirteenth International Conference on Learning Representations*, 2025.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Lopez-Paz, D. and Oquab, M. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Lorden, G. Procedures for reacting to a change in distribution. *The annals of mathematical statistics*, pp. 1897–1908, 1971.

Lu, P., Chen, B., Liu, S., Thapa, R., Boen, J., and Zou, J. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.

Lù, X. H., Kazemnejad, A., Meade, N., Patel, A., Shin, D., Zambrano, A., Stańczak, K., Shaw, P., Pal, C. J., and Reddy, S. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*, 2025.

Mohri, C. and Hashimoto, T. Language models with conformal factuality guarantees. *arXiv preprint arXiv:2402.10978*, 2024.

Narayanan, S., Braza, J. D., Griffiths, R.-R., Ponnapati, M., Bou, A., Laurent, J., Kabeli, O., Wellawatte, G., Cox, S., Rodriques, S. G., et al. Aviary: training language agents on challenging scientific tasks. *arXiv preprint arXiv:2412.21154*, 2024.

Park, Y.-J., Greenewald, K., Alim, K., Wang, H., and Azizan, N. Know what you don't know: Uncertainty calibration of process reward models. *arXiv preprint arXiv:2506.09338*, 2025.

Podkopaev, A. and Ramdas, A. Tracking the risk of a deployed model and detecting harmful distribution shifts. *arXiv preprint arXiv:2110.06177*, 2021.

Prinster, D., Han, X., Liu, A., and Saria, S. Watch: Adaptive monitoring for ai deployments via weighted-conformal martingales. In *Forty-second International Conference on Machine Learning*, 2025.

Qu, Y., Huang, K., Yin, M., Zhan, K., Liu, D., Yin, D., Cousins, H. C., Johnson, W. A., Wang, X., Shah, M., Altman, R. B., Zhou, D., Wang, M., and Cong, L. CRISPR-GPT for agentic automation of gene-editing experiments. *Nat. Biomed. Eng.*, pp. 1–14, July 2025.

Quach, V., Fisch, A., Schuster, T., Yala, A., Sohn, J. H., Jaakkola, T. S., and Barzilay, R. Conformal language modeling. *arXiv preprint arXiv:2306.10193*, 2023.

Ramdas, A. and Manole, T. Randomized and exchangeable improvements of markov's, chebyshev's and chernoff's inequalities. *Statistical Science*, 41(1):121–142, 2026.

Ramdas, A. and Wang, R. Hypothesis testing with e-values. *arXiv preprint arXiv:2410.23614*, 2024.

Schirmer, M., Jazbec, M., Naesseth, C. A., and Nalisnick, E. Monitoring risks in test-time adaptation. *arXiv preprint arXiv:2507.08721*, 2025.

Shafer, G. and Vovk, V. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

Shanmugam, D., Sadhuka, S., Raghavan, M., Guttag, J., Berger, B., and Pierson, E. Evaluating multiple models using labeled and unlabeled data. *arXiv preprint arXiv:2501.11866*, 2025.

Shin, J., Ramdas, A., and Rinaldo, A. E-detectors: A nonparametric framework for sequential change detection. *The New England Journal of Statistics in Data Science*, 2023.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Swanson, K., Wu, W., Bulaong, N. L., Pak, J. E., and Zou, J. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, pp. 1–3, 2025.

Timans, A., Verma, R., Nalisnick, E., and Naesseth, C. A. On continuous monitoring of risk violations under unknown shift. *arXiv preprint arXiv:2506.16416*, 2025.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Ville, J. Jean ville, étude critique de la notion de collectif. 1939.

Vovk, V. and Wang, R. E-values: Calibration, combination and applications. *The Annals of Statistics*, 49(3):1736–1754, 2021.

Vovk, V. and Wang, R. Confidence and discoveries with e-values. *Statistical Science*, 38(2):329–354, 2023.

Vovk, V., Petej, I., Nouretdinov, I., Ahlberg, E., Carlsson, L., and Gammerman, A. Retrain or not retrain: Conformal test martingales for change-point detection. In *Conformal and Probabilistic Prediction and Applications*, pp. 191–210. PMLR, 2021.

Wald, A. and Wolfowitz, J. Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics*, pp. 326–339, 1948.

Wang, H. and Ramdas, A. Anytime-valid t-tests and confidence sequences for gaussian means with unknown variance. *Sequential Analysis*, 44(1):56–110, 2025.

Wang, H., He, Y., Coelho, P. P., Bucci, M., Nazir, A., Chen, B., Trinh, L., Zhang, S., Huang, K., Chandrasekar, V., et al. Spatialagent: An autonomous ai agent for spatial biology. *bioRxiv*, pp. 2025–04, 2025.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024a.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024b.

Wasserman, L., Ramdas, A., and Balakrishnan, S. Universal inference. *Proceedings of the National Academy of Sciences*, 117(29):16880–16890, 2020.

Waudby-Smith, I., Kennedy, E. H., and Ramdas, A. Distribution-uniform anytime-valid sequential inference. *arXiv preprint arXiv:2311.03343*, 2023.

Waudby-Smith, I., Wu, L., Ramdas, A., Karampatziakis, N., and Mineiro, P. Anytime-valid off-policy inference for contextual bandits. *ACM/IMS Journal of Data Science*, 1 (3):1–42, 2024.

Welinder, P., Welling, M., and Perona, P. A lazy man's approach to benchmarking: semisupervised classifier evaluation and recalibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3262–3269, 2013.

Wu, M., Zhou, C., Bates, S., and Jaakkola, T. Thought calibration: Efficient and confident test-time scaling. *arXiv preprint arXiv:2505.18404*, 2025.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Ye, F., Yang, M., Pang, J., Wang, L., Wong, D., Yilmaz, E., Shi, S., and Tu, Z. Benchmarking llms via uncertainty quantification. *Advances in Neural Information Processing Systems*, 37:15356–15385, 2024.

You, W., Xue, A., Havaldar, S., Rao, D., Jin, H., Callison-Burch, C., and Wong, E. Probabilistic soundness guarantees in llm reasoning chains. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7517–7536, 2025.

Zheng, C., Zhang, Z., Zhang, B., Lin, R., Lu, K., Yu, B., Liu, D., Zhou, J., and Lin, J. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024.

A. Appendix

A.1. Theory

A.1.1. Proof of Proposition 1

Let $\{M_t\}_{t\in\mathbb{N}_+}$ denote any scoring process where $M_t = f_t(\mathbf{S}_{[1:t]})$ for a sequence of (deterministic) functions $(f_t)_{t\in\mathbb{N}_+}$ and $M_t \in \mathbb{R}$. For fixed error level $\delta \in (0,1)$ and quantile level $\alpha \in (0,1)$, let c_α be the output of Algorithm 1. Then,

$$\Pr_{\mathcal{D}_{cal}}\left(\Pr_{\mathcal{H}_N}\left(\exists\, t \in [T] : M_t \geq c_\alpha \mid \mathcal{D}_{\text{cal}}\right) \leq \alpha\right) \geq 1-\delta$$

Proof. Suppose our calibration set contains n successful trajectories that are i.i.d. according to the null. Note that $\Pr_{\mathcal{H}_N}(\nexists t : M_t \geq c_\alpha | \mathcal{D}_{\text{cal}}) = \Pr_{\mathcal{H}_N}(\max_{t\in\mathbb{N}} M_t \leq c_\alpha | \mathcal{D}_{\text{cal}})$, since the event that the entire sequence $(M_t)_{t=0}^{\infty}$ is below c_α is equivalent to the event that the maximum is below c_α. Thus, it suffices to consider the maximum score over all steps.

Compute the calibration set maxima $M^{(1)}, \ldots, M^{(n)}$. Note that these maxima, $M^{(i)}, i = 1, \ldots, n$ are i.i.d. from some distribution with (unknown) CDF, F. Define the $(1-\alpha)$-quantile of this distribution as

$$q_{1-\alpha} := \inf\{x \in \mathbb{R} : F(x) \geq 1-\alpha\},$$

such that $F(q_{1-\alpha}-) \leq 1-\alpha \leq F(q_{1-\alpha})$, where $F(q-) := \lim_{x\to q^-} F(x)$. Our goal is to use the calibration data to construct a $(1-\delta)$-confidence upper bound on $q_{1-\alpha}$. That is, we will find c_α such that

$$\Pr_{\mathcal{D}_{\text{cal}}}(c_\alpha < q_{1-\alpha}) \leq \delta. \tag{4}$$

Then, on the event $\{c_\alpha \geq q_{1-\alpha}\}$, we have that

$$\Pr_{\mathcal{H}_N}\left(\max_{t\in\mathbb{N}} M_t \geq c_\alpha \mid \mathcal{D}_{\text{cal}}\right) \leq \Pr_{\mathcal{H}_N}\left(\max_{t\in\mathbb{N}} M_t \geq q_{1-\alpha} \;\middle|\; \mathcal{D}_{\text{cal}}\right) = 1 - F(q_{1-\alpha}-) \leq 1-(1-\alpha) = \alpha. \tag{5}$$

Since by construction, the event $\{c_\alpha \geq q_{1-\alpha}\}$ occurs with probability at least $1-\delta$, we have

$$\Pr_{\mathcal{D}_{\text{cal}}}\left(\Pr_{\mathcal{H}_N}\left(\exists t : \hat{M}_t \geq c_\alpha | \mathcal{D}_{\text{cal}}\right) \leq \alpha\right) = \Pr_{\mathcal{D}_{\text{cal}}}\left(\Pr_{\mathcal{H}_N}\left(\max_{t\in\mathbb{N}} \hat{M}_t \geq c_\alpha \mid \mathcal{D}_{\text{cal}}\right) \leq \alpha\right) \geq 1-\delta \tag{6}$$

as desired.

We construct a $(1-\delta)$-confidence upper bound on $q_{1-\alpha}$ using the following argument, which follows ideas from Clopper & Pearson (1934). Let $M_{(1)} \leq M_{(2)} \leq \cdots \leq M_{(n)}$ denote the order statistics of the calibration score maxima, where ties occupy successive ranks. We will set c_α to be one of these order statistics, as follows. Denote $K(x) := \#\{i : M^{(i)} < x\}$. For any $x \in \mathbb{R}$, $K(x) \sim \text{Binomial}(n, F(x-))$. Then for any k and x, the events $\{M_{(k)} < x\}$ and $\{K(x) \geq k\}$ are equivalent, so

$$\Pr_{\mathcal{D}_{\text{cal}}}(M_{(k)} < x) = \Pr_{\mathcal{D}_{\text{cal}}}(K(x) \geq k) = \Pr(\text{Binomial}(n, F(x-)) \geq k). \tag{7}$$

In particular, for any k, at the quantile, $q_{1-\alpha}$ we have

$$\Pr_{\mathcal{D}_{\text{cal}}}(M_{(k)} < q_{1-\alpha}) = \Pr(\text{Binomial}(n, F(q_{1-\alpha}-)) \geq k) \leq \Pr(\text{Binomial}(n, 1-\alpha) \geq k), \tag{8}$$

where the inequality holds because $F(q_{1-\alpha}-) \leq 1-\alpha$ and $\Pr(\text{Binomial}(n,q) \geq k$ is nondecreasing in q. Therefore, Algorithm 1 chooses $k^\star = \min\{k : \Pr(\text{Binomial}(n, 1-\alpha) \geq k) \leq \delta\}$ and sets $c_\alpha = M_{(k^\star)}$. This is the smallest order statistic such that Eq. (4) holds. That is, by construction, c_α satisfies

$$\Pr_{\mathcal{D}_{\text{cal}}}(c_\alpha < q_{1-\alpha}) = \Pr_{\mathcal{D}_{\text{cal}}}(M_{(k^\star)} < q_{1-\alpha}) \leq \Pr(\text{Binomial}(n, 1-\alpha) \geq k^\star) \leq \delta, \tag{9}$$

as desired.

□

A.1.2. Proof of log-optimality of density ratio process

Let p_0 and p_1 be the alternative and null densities, respectively. The density ratio process given by $M_t = \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})}$ provides log-optimal growth rate. That is, for any other e-process $(M'_t)_{t=1}^{\infty}$ and stopping time τ, $E_{\mathcal{H}_A}[\log M_\tau] \geq E_{\mathcal{H}_A}[\log M'_\tau]$.

Proof. A full proof of this theorem appears as Theorem 7.11 in (Ramdas & Wang, 2024). We refer the reader to that textbook for a rigorous treatment of the proof.

First, note that $(M_t)_{t=1}^{\infty}$ is indeed a test martingale (and thus also an e-process). Let $\mathcal{F}_t = \sigma(\mathbf{S}_{[1:t]})$ be the natural filtration. We show that $(M_t)_{t\in\mathbb{N}}$ satisfies the definition of a test martingale for $\mathcal{H}_N : \mathbf{S} \sim P_1$. First, note that density ratios are always non-negative, so M_t is always non-negative. Also, since by construction $M_0 = 1$, we have $\mathbb{E}_{\mathcal{H}_N}[M_0] = 1$. We now show that $(M_t)_{t\in\mathbb{N}}$ is a martingale under $\mathcal{H}_N$, that is, when $\mathbf{S} \sim P_1$. We have

$$
\begin{aligned}
\mathbb{E}_{\mathcal{H}_N}[M_t \mid \mathcal{F}_{t-1}] &= \mathbb{E}_{\mathcal{H}_N}\left[\frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})} \,\middle|\, \mathcal{F}_{t-1}\right] \\
&= \frac{p_0(\mathbf{S}_{[1:t-1]})}{p_1(\mathbf{S}_{[1:t-1]})} \int \frac{p_0(s_t \mid \mathbf{S}_{[1:t-1]})}{p_1(s_t \mid \mathbf{S}_{[1:t-1]})} p_1(s_t \mid \mathbf{S}_{[1:t-1]})\, ds_t \\
&= M_{t-1} \int p_0(s_t \mid \mathbf{S}_{[1:t-1]})\, ds_t \\
&= M_{t-1}
\end{aligned}
$$

where the third equality holds because the integral of a density is 1. Thus, the process $(M_t)_{t\in\mathbb{N}}$ is a test martingale for the null hypothesis.

The main log-optimality result is an extension of the fact that in a non-sequential hypothesis test of P_1 against P_0, where $P_0 \ll P_1$, the likelihood ratio, $E = p_0/p_1$, is the log-optimal e-variable: $\mathbb{E}_{P_0}[\log E'] \leq \mathbb{E}_{P_0}[\log E]$ for any other e-variable E' for P_1. To see this, first note that it suffices to consider e-variables of the form $E' = dQ/dP_1$ for distributions Q such that $Q \ll P_0 \ll P_1$. We have that

$$
\mathbb{E}_{\mathcal{H}_A}\left[\log \frac{E'}{E}\right] = \int p_0(x) \log\left(\frac{q(x)/p_1(x)}{p_0(x)/p_1(x)}\right) L(dx) = -\int p_0(x) \log\left(\frac{p_0(x)}{q(x)}\right) L(dx) \leq 0, \tag{10}
$$

where L is the reference measure that the densities p_1, p_0, q are defined with respect to. That is, $\mathbb{E}_{\mathcal{H}_A}[\log E'] \leq \mathbb{E}_{\mathcal{H}_A}[\log E]$.

From here, one can extend this statement into the sequential setting of e-processes, which is done in Theorem 7.11 of the reference. □

A.1.3. Density ratio estimation procedure

Algorithm 2 Density ratio estimation using calibration data.

Inputs: calibration data $\mathcal{D}_{\text{cal}}$.
Output: functions that estimate the density ratio for every step, $\{\hat{M}_t\}_{t \in \mathbb{N}_+}$.
Split $\mathcal{D}_{\text{cal}}$ randomly, where $\mathcal{D}_{\text{DRE}} \cup \mathcal{D}_{\text{threshold}} = \mathcal{D}_{\text{cal}}$.
$\hat{\pi}_1 \leftarrow \frac{1}{|\mathcal{D}_{\text{DRE}}|} \sum_{(\mathbf{S},Y) \in \mathcal{D}_{\text{DRE}}} Y$ ▷ Estimate class priors.
$T_{\text{max}} \leftarrow \max\{T : Y \in \{0,1\}$ both in $\mathcal{D}_{\text{DRE}}$ at len $T\}$
for $t = 1, \ldots, T_{\text{max}}$ **do**
  $\mathcal{D}_{\text{DRE},t} \leftarrow \{(\mathbf{S}_{[1:t]}, Y) : (\mathbf{S}, Y) \in \mathcal{D}_{\text{DRE}} : |\mathbf{S}| \geq t\}$
  Use $\mathcal{D}_{\text{DRE},t}$ to train probabilistic classifier $\hat{g}_t$ that takes $\mathbf{S}_{[1:t]}$ as input and predicts $p(Y = 1 \mid \mathbf{S}_{[1:t]})$.
  $\hat{M}_t(\cdot) \leftarrow \frac{1-\hat{g}_t(\cdot)}{\hat{g}_t(\cdot)} \frac{\hat{\pi}_1}{1-\hat{\pi}_1}$
end for
Set $\hat{M}_t(\cdot) \leftarrow \hat{M}_{T_{\text{max}}}(\cdot)$ for all $t > T_{\text{max}}$.

Algorithm 2 specifies our density ratio estimation procedure. Note that we train a separate density ratio estimator per timestep t.

A.1.4. Baseline: Randomized Ville

(Ramdas & Manole, 2026) provide an alternative procedure for controlling FAR while increasing power in sequential testing using a *randomized* variant of Ville's inequality, which motivates our use of it as a baseline. We refer to their paper for a detailed treatment of the procedure but provide the main result here for the sake of clarity.

Proposition A.1. *(Randomized Ville's inequality) Let $(M_t)_{t\geq 0}$ be any non-negative supermartingale with respect to a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$. Let $E[M_0] \leq 1$. Then, for any $\mathbb{F}$-stopping time τ:*

$$Pr\left[\exists t < \tau : M_t \geq \frac{1}{\alpha} \text{ OR } M_\tau \geq \frac{Z}{\alpha}\right] \leq \alpha$$

where Z is super-uniform on $[0, 1]$ and independent of $\mathbb{F}$ and $\alpha \in (0, 1)$.

Proof. See (Ramdas & Manole, 2026) for the proof. □

The implication of the above result is that for a (perfectly estimated) density ratio process $M_t = \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})}$ (see Eq. 2), one can control FAR *and* terminate by time T (the end of the trajectory) by *sampling* the threshold Z at any $\mathbb{F}$-adapted stopping time τ. Z can be sampled from any super-uniform distribution on $[0, 1]$, i.e., Z can be sampled from any distribution whose CDF satisfies $P(Z \leq u) \leq u$ for all $u \in [0, 1]$. Furthermore, the test *ends* after τ and the user may not continue testing after this point.

We experimented with this approach in our randomized Ville baseline, setting the stopping time τ as the final step T in the trajectory, and sampling $Z \sim \text{Unif}(0, 1)$.

A.1.5. Toy example: Marginal calibration does not control false alarm rate

A verifier score, S, satisfies *marginal calibration* if

$$p(Y = 1 \mid S) = S \tag{11}$$

almost surely. Although a commonly used notion of probabilistic "correctness", marginal calibration does not enable control of the false alarm rate, as it does not account for the base rate of the null and alternative (i.e., $p(Y = 1)$ and $p(Y = 0)$). This is the case in the sequential hypothesis setting, where marginal calibration of S_t at each step t does not enable anytime control of the false alarm rate, but it is also the case in the simple non-sequential setting, as illustrated by the following example.

Suppose we have a verifier score, $S \in [0, 1]$, that only takes on two values: $S \in \{0.005, 0.5\}$, with $p(S = 0.005) = 0.99$ and $p(S = 0.5) = 0.01$. The verifier score is marginally calibrated, so $p(Y = 1 \mid s = 0.005) = 0.005$ and $p(Y = 1 \mid s = 0.5) = 0.5$. As a naive attempt to control the false alarm rate, such that $p(\texttt{reject} \mid Y = 1) \leq \alpha = 0.01$, we decide to `reject` whenever $S \leq \alpha = 0.01$. We now calculate the resulting false alarm rate.

First, the base rate of the null is

$$\begin{aligned} p(Y = 1) &= p(Y = 1 \mid S = 0.005) \cdot p(S = 0.005) + p(Y = 1 \mid S = 0.5) \cdot p(S = 0.5) \\ &= 0.005 \cdot 0.99 + 0.5 \cdot 0.01 \\ &= 0.00995. \end{aligned}$$

The false alarm rate is $p(\text{reject} \mid Y = 1)$, which is equivalent to $p(S = 0.005 \mid Y = 1)$ since we reject for $S \leq 0.01$. However,

$$\begin{aligned} p(\text{reject} \mid Y = 1) &= p(S = 0.005 \mid Y = 1) \\ &= \frac{p(Y = 1 \mid S = 0.005)\, p(s = 0.005)}{p(Y = 1)} \\ &= \frac{0.005 \cdot 0.99}{0.00995} \\ &\approx 0.50 \gg 0.01. \end{aligned}$$

Thus, even with a marginally calibrated verifier score, S, rejecting the null when $p(Y = 1 \mid S) \leq \alpha$ does not control the false alarm rate.

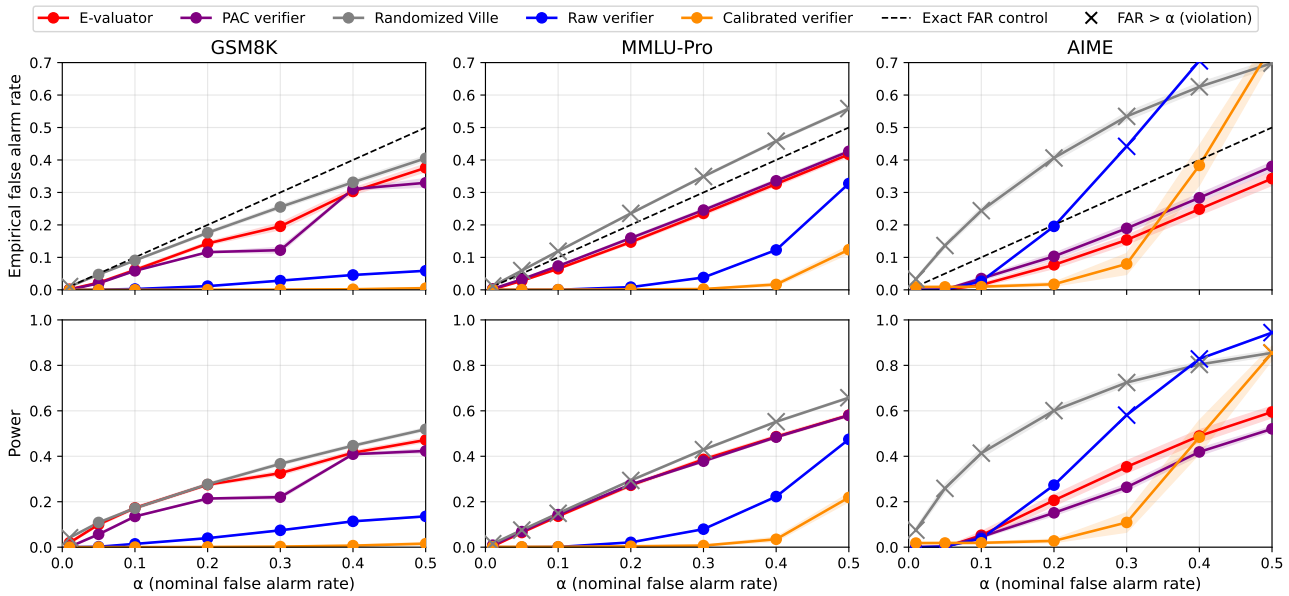

Figure 4. **GSM8k, MMLU-Pro, and AIME results**. The false alarm rate is empirically controlled for *e-valuator* and the PAC verifier. Additionally, *e-valuator* achieves greater (or equal) power among methods that are able to control the false alarm rate.

A.2. Additional results

A.2.1. Additional false alarm rate and power results

We provide false alarm rate and power results from three additional datasets not presented in the main text (Section 4.1): GSM8k (Cobbe et al., 2021), MMLU-Pro (Wang et al., 2024b), and AIME (**?**) (Figure 4). *E-valuator* empirically controls the false alarm rate for all choices of α. The PAC verifier also controls the false alarm rate; however, it has worse (or equal) power on the datasets shown. Randomized Ville is able to control the false alarm rate on GSM8k but not on MMLU-Pro and AIME.

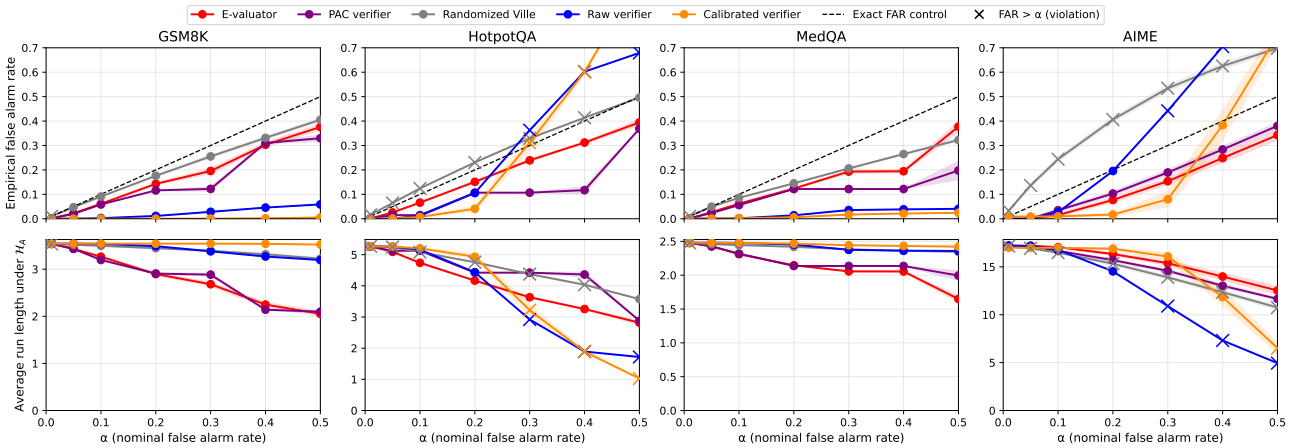

Figure 5. **Average run length results**. The average run length for unsuccessful trajectories is generally lower for e-valuator than other methods, indicating that it is able to detect unsuccessful trajectories earlier. Violations of FAR control are marked with an X.

A.2.2. Additional average run length under alternative results

We provide average run length under alternative results from four additional datasets not presented in the main text: GSM8k (Cobbe et al., 2021), HotpotQA (Yang et al., 2018), MedQA (Jin et al., 2021), and AIME (**?**) (Figure 5). E-valuator provides better average run length in all but one dataset (AIME). For instance, for HotpotQA, at $\alpha = 0.4$, e-valuator detects unsuccessful trajectories within 3.2 steps on average, whereas the PAC verifier uses 4.3 steps. The raw verifier and randomized Ville baselines do not control FAR in general.

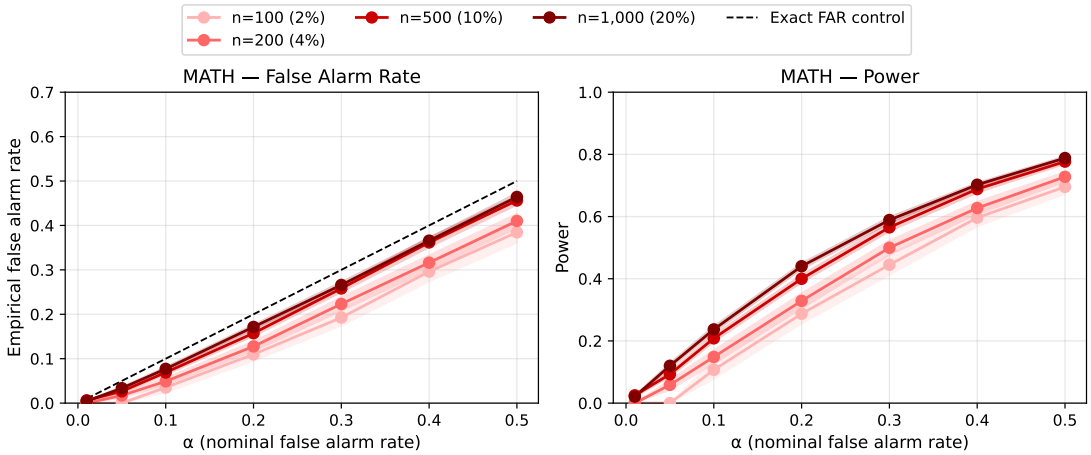

Figure 6. **Calibration set size**. On the MATH dataset, the false alarm rate is empirically controlled for most calibration set sizes. At very small sizes (2% of the data, or 100 labeled trajectories), *e-valuator* can fail to control the false alarm rate (at very low α). However, in general, the false alarm rate is controlled across α and calibration set sizes.

A.2.3. Ablations of calibration set size

We examine the effect of the size of the calibration set on *e-valuator*. Recall that we split $\mathcal{D}_{\text{cal}}$ into $\mathcal{D}_{\text{DRE}}$ and $\mathcal{D}_{\text{threshold}}$, learning the density ratios on the former split, and estimating the rejection threshold on the latter. Because our density ratios are learned, we expect these to ratios to be more accurate as the calibration set size increases.

We run this ablation on the MATH dataset, which has 5000 total trajectories. As shown in Figure 6, the size of the calibration set has little effect on the empirical false alarm rates and power. However, at very small amounts of calibration data (2%, or 100 labeled trajectories), the density ratios tend to be noisier, leading to greater variance in the false alarm rate and power.

We observe that the false alarm rates remain similar as the calibration set size increases (and all sizes control the false alarm rate).

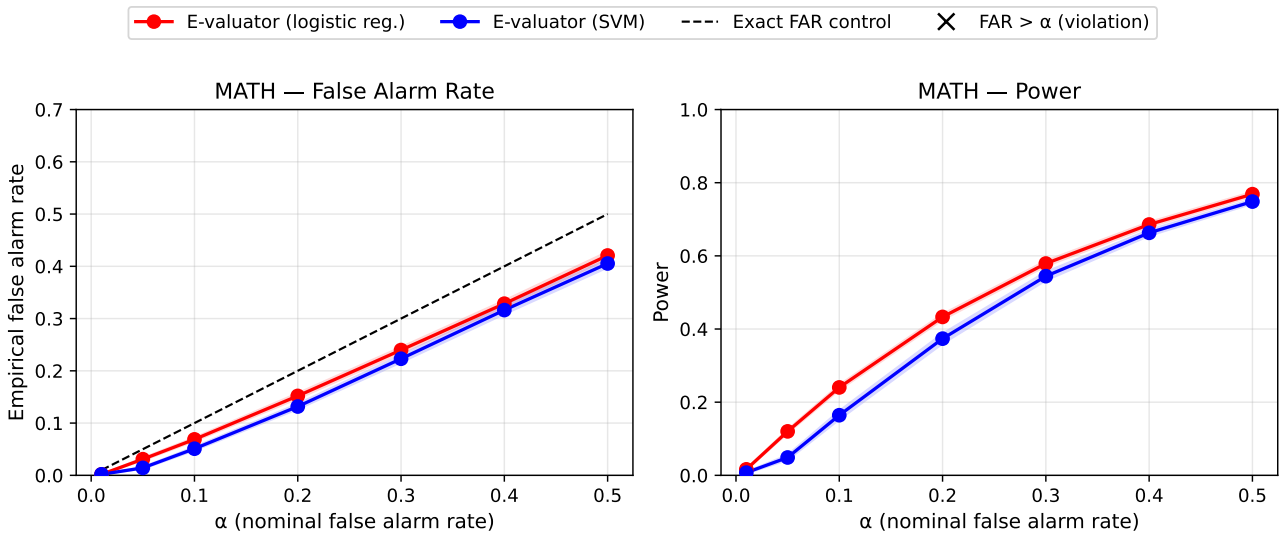

Figure 7. **Density ratio estimator**. Changing the density ratio estimator from logistic regression to SVM does not substantially change the results on the MATH dataset. False alarm rate control and power track closely across both density ratio estimation methods.

A.2.4. Ablations of density ratio estimators

We examine the effect of the density ratio estimation method on *e-valuator*. Here we use a support vector machine instead of logistic regression as the density ratio estimation method. On the MATH dataset, we find that the false rate and power track closely across both methods (Figure 7).

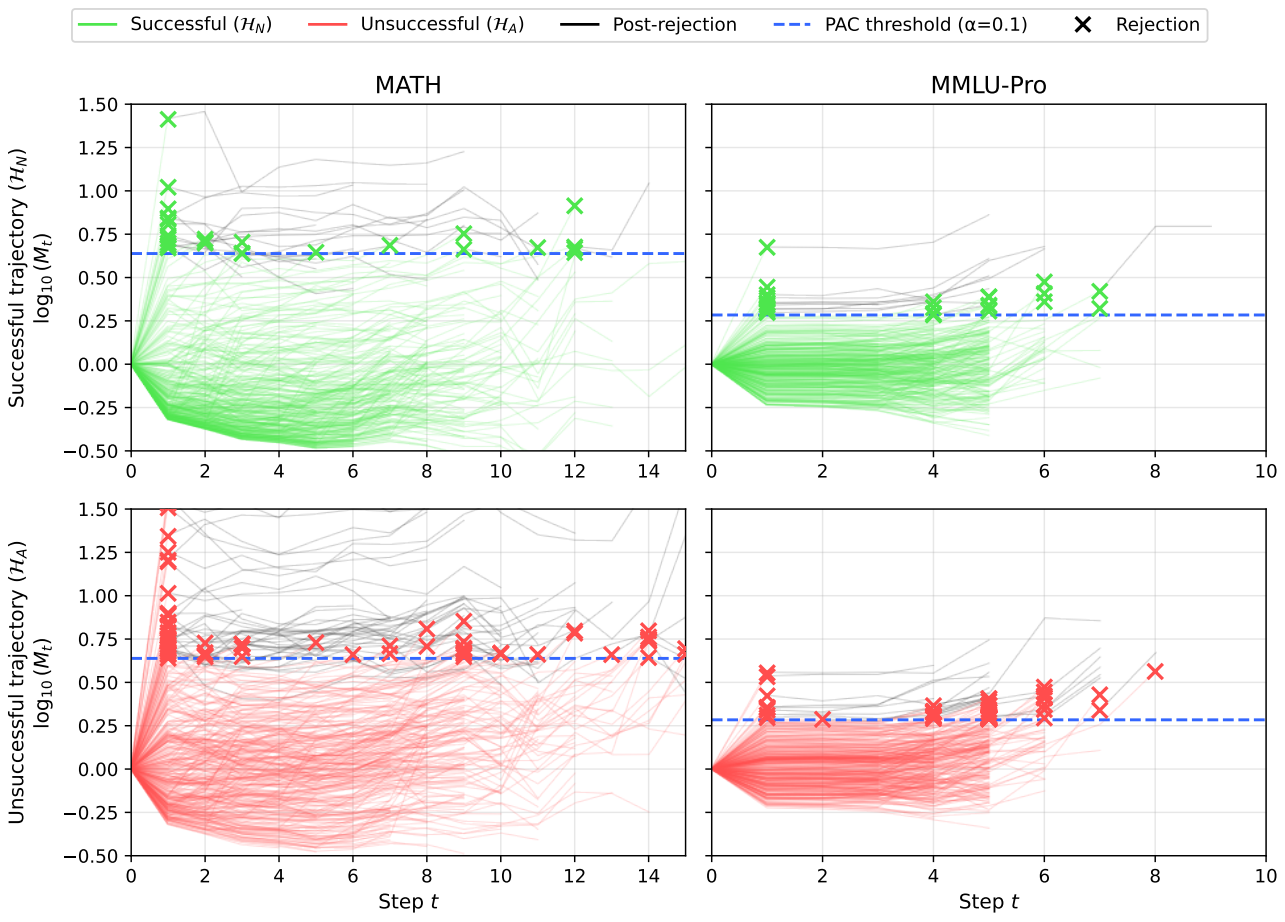

Figure 8. **Example sequences**. On the MATH dataset (left column), many of the rejections are generated at M_1, after the first action, indicating the first action of the agent is important in deciding success. On the MMLU-Pro dataset (right column), which has shorter trajectories, there are more rejections later in the sequence.

A.2.5. Example M_t sequences

We additionally show some example M_t sequences (plotted as $\log(M_t)$ for visual clarity) in Figure 8 for the MATH and MMLU-Pro datasets. We observe that for both datasets, $\log(M_t)$ inflates above c_α for unsuccessful trajectories ($\mathcal{H}_A$). By contrast, few M_t sequences cross the threshold c_α among successful trajectories. Nonetheless, there is visual heterogeneity in the sequences. In general, we expect more powerful verifiers/PRMs to provide stronger visual separation of $\mathcal{H}_N$ and $\mathcal{H}_A$ trajectories.

A.3. Details on datasets, agents, and verifiers

We provide experiments from six different datasets. For each dataset, we use a particular agent-verifier combination, which we list in Table A.3.

For **mathematical reasoning**, we use GSM8k (Cobbe et al., 2021) for our tool-calling agent experiments and MATH (Hendrycks et al., 2021) and AIME (**?**) for our reasoning model experiments. For **question-answering**, we use HotpotQA (Yang et al., 2018) and MedQA (Jin et al., 2021) for Aviary and OctoTools experiments, respectively, and MMLU-Pro (Wang et al., 2024b) for the reasoning models. We present results from all datasets except GSM8k and AIME in the main section, and provide results for these two datasets in the Appendix.

For tool-calling agents, we provide the verifier (a judge LLM) with the original problem text and list of tool calls and arguments used. We then prompt the Claude agent with the following system prompt:

> You are an expert in analyzing agent trajectories and estimating the probability of success. Your final answer should be of the form: `[PROBABILITY]: [number between 0 and 1]`
>
> The probability value should be a number between 0 and 1. LIMIT YOUR RESPONSE TO JUST `[PROBABILITY]: [number between 0 and 1]`, or else I will switch to OpenAI.
>
> The agent in question is a LLM-based agent that uses tools to solve problems. The agent may not make more than {`max_tool_calls`} total tool calls. If it does, it will be terminated with an error.
>
> The tools this agent can use are:
>
> - `submit_answer`
> - `search`
> - `lookup`
>
> You will be given a partial trajectory of the agent's actions. Your task is to estimate the probability of success of the agent given the partial trajectory.
>
> Your probability should incorporate the following:
>
> - The tools that the agent has used
> - The arguments that the agent has used, including the syntax of the arguments
> - The problem text
> - The number of total tool calls allowed
>
> Here is the final answer format: `[PROBABILITY]: [number between 0 and 1]`
>
> {`partial_trajectory`}

For the reasoning model, we simply provide the pretrained process reward model (Wang et al., 2024a) the reasoning trace and it outputs a logits-based probability that the trajectory is successful after each step.

Table A.3 lists all the datasets, agents, and verifiers used. Table 2 additional lists some statistics on the quality of these verifiers. As shown, there is substantial variation in the verifier quality. For instance, on MedQA, the verifier has strong discriminative power (AUC 0.7) and is well-calibrated (ECE 0.021). By contrast, on MATH, the verifier has strong predictive power (AUC 0.692) but is poorly calibrated (ECE 0.184). Finally, on GSM8k, the verifier has weak predictive power (AUC 0.602) but is reasonably calibrated (ECE 0.053).

Dataset	Domain	Agent	Verifier	Agent Description
GSM8k (Cobbe et al., 2021)	Math reasoning	Aviary (Narayanan et al., 2024)	Claude Haiku 3.5	Tool-calling agent for math QA. Text-based verifier model.
MATH (Hendrycks et al., 2021)	Math reasoning	Claude Sonnet 4	Pretrained PRM (Wang et al., 2024a)	Multi-step reasoning model, with pretrained verifier model.
AIME (**?**)	Math reasoning	Claude Sonnet 4	Pretrained PRM (Wang et al., 2024a)	Multi-step reasoning agent, with pretrained verifier model.
HotpotQA (Yang et al., 2018)	QA	Aviary	Claude Haiku 3.5	Tool-calling agent for general QA. Text-based verifier model.
MedQA (Jin et al., 2021)	QA	OctoTools (Lu et al., 2025)	Claude Haiku 3.5	Tool-calling agent for medical QA. Text-based verifier model.
MMLU-Pro (Wang et al., 2024b)	QA	Claude Sonnet 4	Pretrained PRM (Wang et al., 2024a)	Multi-step reasoning agent, with pretrained verifier model.

Table 1. List of datasets, agents, and verifiers used in our experiments.

Dataset	Verifier	Verifier AUC	Verifier ECE
MedQA	Claude Haiku 3.5	0.700	0.021
MATH	Pretrained PRM	0.692	0.184
HotpotQA	Claude Haiku 3.5	0.654	0.137
AIME	Pretrained PRM	0.624	0.083
GSM8k	Claude Haiku 3.5	0.602	0.053
MMLU-Pro	Pretrained PRM	0.592	0.086

Table 2. Verifier quality across datasets, measured by AUC and expected calibration error (ECE), both averaged across all timesteps and trajectories.

A.3.1. Additional computational details

We use the default hyperparameter settings in `scikit-learn` with logistic regression for all experiments presented in this paper. Given a set of verifier scores, all experiments in this paper can be completed in under a minute on a standard laptop.