E-VALUATOR: RELIABLE AGENT VERIFIERS WITH SEQUENTIAL HYPOTHESIS TESTING

Anonymous authors
Paper under double-blind review

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 through sequences of mechanical actions (Bekey, 1998) or game-playing agents that play games through sequences of game-legal actions (Brown and Sandholm, 2019; Silver et al., 2017). Agents have wide applications, and have delivered promising results across diverse domains, including identifying drug candidates to target 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 finetuned to give a prediction of whether each step in a trajectory is "correct" or "incorrect" (Li and Li, 2025; Zheng et al., 2025; 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

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

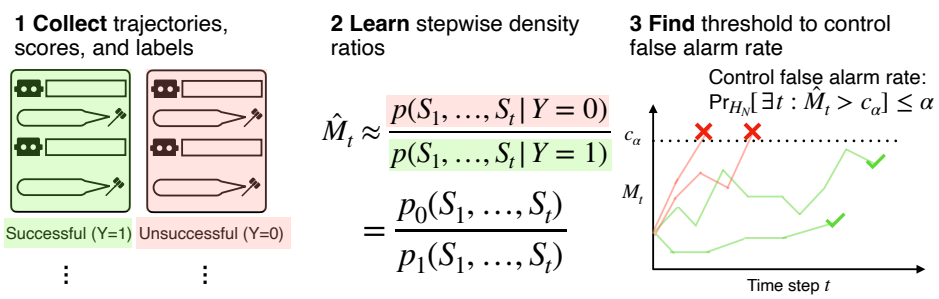

Figure 1: **E-valuator overview**. *E-valuator* works in three steps. First, we collect a small calibration set of trajectories, verifier scores, and labels. Second, we learn the density ratios $\hat{M}_t$ at each timestep t. Third, we find a decision threshold that controls the false alarm rate with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

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 (Guo et al., 2017), such notions do not provide guarantees on the false alarm rate. Furthermore, each action in a trajectory costs time and resources, 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 continuously assessing the verifier score (e.g., after each action), a 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 (see Figure 1). 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 and Wang, 2025; Wald and Wolfowitz, 1948) to construct a specific sequential process atop the black-box verifier scores—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.

Importantly, *e-valuator* complements any future improvements to verifiers: while it guarantees control of the false alarm rate for any agent and verifier, deploying it with better verifiers will tend to yield decision rules with higher power—that is, higher rates of correctly flagging unsuccessful trajectories. *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, calibrated, or PAC thresholded verifier alone. We also show that e-valuator terminates unsuccessful trajectories earlier than these baselines, which enables more favorable tradeoffs between token usage and accuracy. For instance, e-valuator recovers up to 90% of an agent's original accuracy with just 80% of the tokens on one dataset tested.

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

Sequential hypothesis test. 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 corresponding 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, $\mathcal{H}_N$, 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, $\alpha \in (0, 1)$.

False alarm rate control. For $t = 1, 2, \ldots$, if $M_t > c_\alpha$, 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[2]

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

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).

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,

[1] If $P_1 = P_0$, the score sequences would be identically distributed regardless of whether the trajectory is successful or not, meaning the verifier scores carry no information about correctness. In this case, no test based on the scores can do better than random guessing.

[2] One could instead control the missed detection rate, i.e., the probability of failing to flag an unsuccessful trajectory as unsuccessful. In this case, one would swap $\mathcal{H}_N$ and $\mathcal{H}_A$ and apply the same techniques. We leave the choice of which error rate to control to the user.

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.
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 fair coin
$k \leftarrow \min\{j \in [n] : \Pr[\text{Bin}(n, 1-\alpha) \geq j] \leq \delta\}$ ▷ Find index using Binomial tail bound
$c_\alpha \leftarrow M_{(k)}$

$\mathcal{D}_{\text{cal}}$, 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* 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 $M_t \in \mathbb{R}$ (e.g., f_t could be the identity function, $M_t = f_t(\mathbf{S}_{[1:t]}) = S_t$). 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 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 ("probably"), the procedure controls of the false alarm rate under a user-specified α ("approximately correct").

Proposition 2.1 (PAC thresholding)**.** *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,*

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

See Appendix A.1 for the proof. Observe that Proposition 2.1 involves two error parameters: α (the quantile level) and δ (the probability of FAR miscalibration). Since c_α is a function of $\mathcal{D}_{\text{cal}}$, the *marginal* false alarm rate is the conditional FAR averaged over the draw of the calibration set:

$$\Pr_{\mathcal{H}_N}[\exists\, t \in [T] : M_t \geq c_\alpha] = \mathbb{E}_{\mathcal{D}_{\text{cal}}}\left[\Pr_{\mathcal{H}_N}\left(\exists\, t \in [T] : M_t \geq c_\alpha \mid \mathcal{D}_{\text{cal}}\right)\right] = \mathbb{E}_{\mathcal{D}_{\text{cal}}}\left[\text{FAR}(c_\alpha)\right],$$

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 $\alpha + \delta \leq \alpha_0$. The choice of α' and δ is constrained by the number of samples with $Y = 1$ the threshold calibration set, denoted n. If n is too small, the algorithm returns $c_\alpha = \infty$ and the procedure never rejects (trivially controlling FAR but obtaining zero power), which occurs when $n < \log\delta / \log(1-\alpha')$. Thus, we recommend setting α', δ such that $n \geq \lceil \log\delta / \log(1-\alpha') \rceil$.

2.4 Designing a (log-)optimal test statistic

Although Proposition 2.1 controls the false alarm rate with high probability for any function $(f_t)_{t\in\mathbb{N}_+}$, it does not guide *which* functions we should use. Our test rejects a trajectory when M_t first exceeds c_α. Thus, an ideal $(f_t)_{t\in\mathbb{N}_+}$ will have M_t grow rapidly under $\mathcal{H}_A$ (when the trajectory is unsuccessful).

Test martingales. To do so, we take inspiration from *test martingales* (Ramdas and Wang, 2025), although our final statistic is not an exact test martingale. Test martingales are a framework for sequential hypothesis testing that enables principled construction of $(f_t)_{t\in\mathbb{N}}$ and additional guarantees. A test martingale for $\mathcal{H}_N$ is a sequence, $(M_t)_t$, that satisfies: (a) **Non-negativity and unit mean**: M_t is non-negative for all t and $E_{\mathcal{H}_N}[M_0] \leq 1$. (b) **Martingale**. $(M_t)_t$ is a martingale for $\mathcal{H}_N$. For all t, $\mathbb{E}_{\mathcal{H}_N}[M_t \mid M_0, \ldots, M_{t-1}] = M_{t-1}$.

Test martingales 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 test martingale, 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 test martingale. Test martingales are a special case of a larger set of sequential hypothesis tests known as e-processes, which we do not discuss here.

Density ratio is log-optimal. 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 is log-optimal, meaning it 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 for a formal statement and proof of this. Intuitively, M_t will tend to surpass the decision threshold, and correspondingly enable detection of unsuccessful trajectories earlier than other test statistics.

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 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 (Dandapanthula and Ramdas, 2025).

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 and Hyvärinen, 2012):

$$M_t = f_t(\mathbf{S}_{[1:t]}) = \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})} = \frac{p(T \geq t, \mathbf{S}_{[1:t]} \mid Y = 0)}{p(T \geq t, \mathbf{S}_{[1:t]} \mid Y = 1)} \tag{3}$$

$$= \frac{p(Y = 0 \mid T \geq t, \mathbf{S}_{[1:t]})}{p(Y = 1 \mid T \geq t, \mathbf{S}_{[1:t]})} \cdot \frac{p(Y = 1)}{p(Y = 0)}. \tag{4}$$

Thus, for each time step t, we train a classifier, $\hat{g}_t$, on trajectories satisfying $T \geq t$, which takes $\mathbf{S}_{[1:t]}$ as input and provides an estimate of $p(Y = 1 \mid T \geq t, \mathbf{S}_{[1:t]})$. We also form an estimate, $\hat{\pi}_1$, of the unconditional class probability $p(Y = 1)$ from the density-ratio-estimation set $\mathcal{D}_{\text{DRE}}$. We plug these two estimates into Eq. equation 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 B.5).

3 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 for $\mathcal{H}_A$ 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, each in a different setting.

Agents and verifiers. We conduct experiments on two tool-calling agents, Aviary (Narayanan et al., 2025) and Octotools (Lu et al., 2025), and 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 (Veeraboina, 2023)), 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 B. Results for each dataset are from one agent-verifier combination. A full description of the dataset, agent, and verifier combinations is available in Appendix C. 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 *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. Like e-valautor, the **PAC verifier** 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. We also compare to conformal risk control (Angelopoulos et al., 2024), which uses the same trajectory-level raw verifier scores but controls FAR only in expectation instead of with high probability over the draw of the calibration set (and thus not directly comparable), in Appendix B.7. Empirically, this procedure achieves higher power than the PAC verifier at the cost of more frequent FAR violations.
4. **Randomized Ville** utilizes a modern variant of Ville's inequality (Ramdas and 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.3. This procedure guarantees FAR 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 $\alpha' + \delta = \alpha$. We use the same α', δ split for the PAC verifier. We include an ablation of the α', δ split in Appendix . For the results presented in the main section, we use an 80/20 split of

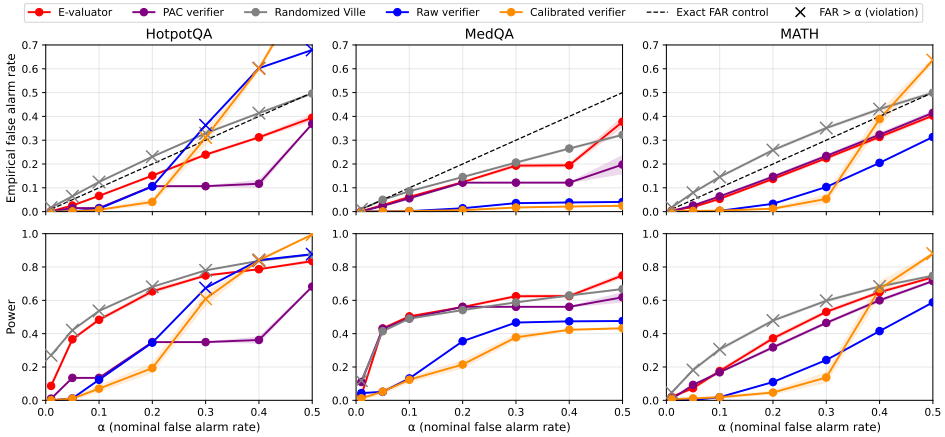

Figure 2: **E-valuator controls the false alarm rate and maximizes power better than alternative methods**. Violations of the false alarm rate control are marked with an X. Randomized Ville, the raw verifier, and the calibrated verifier occassionally provide comparable power, at the cost of inflating 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. The bottom row plots power against the empirical false alarm rate, tracing each method across the same grid of α values. All plots show the 95% CI over 50 random splits of each dataset.

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 B.5 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 B. Finally, we also ablate the density ratio estimator in Appendix B.6.

3.1 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 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 for further discussion). Similarly, randomized Ville 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 (Algorithm 1 applied to the raw verifier scores) is able to control the false alarm rate, as expected. However, it suffers in power, as we discuss below.

3.2 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

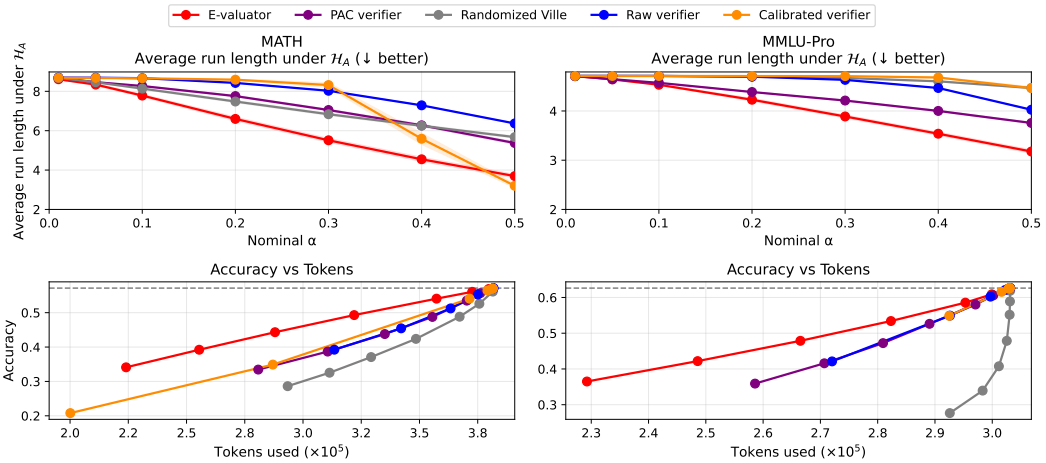

Figure 3: **E-valuator terminates unsuccessful trajectories earlier, providing better token-accuracy tradeoff.** (Top) Average run length under $\mathcal{H}_A$ (↓ better): *e-valuator* rejects unsuccessful trajectories in fewer steps (on average) than competing methods. (Bottom) By terminating unsuccessful 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.

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). The PAC verifier results confirm that PAC thresholding alone (without the density ratio statistic) is insufficient: it controls FAR but achieves lower power, demonstrating that log-optimality is useful for detecting unsuccessful trajectories effectively. In particular, given that PAC verifier and e-valuator both use the same threshold calibration rule (Algorithm 1), the gain in power due to the learned density ratio is precisely e-valuator's improved power over that baseline. Indeed, e-valuator achieves better empirical power at the same empirical FAR than the PAC verifier (and other baselines). These findings hold true across datasets. Similar plots comparing false alarm rate control and power are available in Appendix B for the MMLU-Pro, AIME, and GSM8k datasets.

Distribution shift. E-valuator's guarantees primarily apply to in-distribution trajectories, similar to guarantees in conformal prediction (Angelopoulos et al., 2024; Dandapanthula and Ramdas, 2025) and relaetd literatures. Although e-valuator requires in-distribution null trajectories when calibrating the threshold, it is still robust to certain types of distribution shift. Note that our (learned) density ratio tracks $p(\mathbf{S}|Y=1)/p(\mathbf{S}|Y=0)$. If these two distributions are constant between the calibration and test set–for instance, if $p(Y)$ changes but not $p(\mathbf{S}|Y)$–then our guarantees will still apply. To see this, note that the threshold c_α is chosen only among null samples (Algorithm 1). We empirically validate this finding in Appendix B.3. For other types of shifts that may affect $p(\mathbf{S}|Y)$ in unknown ways, such as shifts in the verifier, agent, or tool environment, e-valuator can no longer provide exact guarantees. Nonetheless, we empirically study e-valuator's robustness to unknown shifts in Appendix B.3 and find that e-valuator is more robust to changes in one component at a time (e.g., just the verifier) than changes in all components (e.g., dataset, agent, verifier) simultaneously.

3.3 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 any 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 $\mathcal{H}_A$. This is practically important: every additional step of an unsuccessful trajectory wastes tokens/compute.

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). Similar to power, the PAC verifier terminates unsuccessful trajectories later than *e-valuator*, further confirming that the density ratio statistic (not just the PAC threshold alone) is important. 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 trajectory 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.

4 Related Work

We provide an abridged overview of related work in this section and give a more thorough treatment in Appendix D.

Verifiers and PRMs. As *e-valuator* is a statistical wrapper for any verifier, our work is relevant to prior work on building better verifiers. These verifiers are often trained as models that estimate a reward (e.g., correctness) 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 not (Uesato et al., 2022; Wang et al., 2024a; Khalifa et al., 2025). Some prior works also calibrate existing PRMs (You et al., 2025; Park et al., 2026), although calibration alone is insufficient to control false alarm rates. Training PRMs can be expensive, as it requires (a) access to process labels (Lightman et al., 2023) and (b) finetuning existing LLMs (Wang et al., 2024a).

Hypothesis testing and AI monitoring. E-valuator takes inspiration from prior work in e-values, which have useful properties for sequential hypothesis testing (Ramdas and Wang, 2025; Vovk and Wang, 2021; 2023). They have found applications in many settings, including A/B testing (Johari et al., 2017), changepoint detection (Shin et al., 2023; Lorden, 1971), and others (Chen and Wang; Huang et al., 2025a; Dhillon et al., 2026). Safe anytime-valid testing has also been applied to track the risk of deployed models Podkopaev and Ramdas (2022) and sequentially test if a classifier is fair (Chugg et al., 2023), among other applications (I Amoukou et al., 2024; Schirmer et al., 2026; Timans et al., 2025). (Jang et al., 2022) applies classifier two-sample tests (Lopez-Paz and Oquab, 2017) to detect covariate shifts.

Statistical guarantees for LLMs Another suite of methods aims to provide statistical guarantees atop LLMs. Conformal prediction methods (Shafer and Vovk, 2008) quantify uncertainty in individual predictions while providing finite-sample and black-box guarantees in those uncertainties. These ideas have been applied to control the factuality of LLM outputs (Cherian et al., 2024; Mohri and 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 but requires white-box access to the LLM's internal logits.

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.

Importantly, as discussed in Section 3.2, e-valuator does not provide guarantees to distribution shifts beyond label shift. Extending e-valuator to accomodate (with guarantees) shifts in the verifier, agent, and dataset between calibration and test time is an important direction of future work. Beyond the applications explored in this paper, one could use *e-valuator* for more nuanced test-time scaling

strategies than the ones explored here, such as resampling bad trajectories. Finally, *e-valuator* can be used for other applications, such as early detection of unsafe or harmful trajectories.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for aid in writing code, generating plots, retrieval and discovery (finding related work) and proving mathematical claims (editing writing/proofs in Appendix). We have not used generative AI tools for generating writing from scratch or proofs from scratch, generating citations, and research ideation and the task of generating synthetic datasets is not applicable to this work. Additionally, we used generative AI tools for editing and generating plots and for managing experiments. We have reviewed all AI-assisted work. We have carefully check the code and plots produced in this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

See the ICLR 2027 AI Policy for Authors for more details. This statement should not be more than 1 page.

ETHICS STATEMENT

E-valuator can be used to stop trajectories for any definition of a “bad" label, including ones that may have nefarious purposes. We, nonetheless, believe, that e-valuator can be used as a good auditing and safety monitoring tool when used correctly.

REPRODUCIBILITY STATEMENT

We included anonymized code as a supplemental file, along with cleaned datasets with verifier scores, from which readers may reproduce our results. Additionally, we have described key hyperparameters (e.g., α', δ split, calibration set size) in the main text of the paper.

REFERENCES

Anastasios Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *International conference on learning representations*, volume 2024, pages 55198–55218, 2024.

Anastasios N Angelopoulos, Stephen Bates, Emmanuel J Candès, Michael I Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, Andre Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K Surikuchi, Ece Takmaz, and Alberto Testoni. LLMs instead of human judges? A large scale empirical study across 20 NLP evaluation tasks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–255. Association for Computational Linguistics, 2025.

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

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

Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

Can Chen and Jun-Kun Wang. Online detection of LLM-generated texts via sequential hypothesis testing by betting. In *Forty-second International Conference on Machine Learning*.

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

Ben Chugg, Santiago Cortes-Gomez, Bryan Wilder, and Aaditya Ramdas. Auditing fairness by betting. *Advances in Neural Information Processing Systems*, 36:6070–6091, 2023.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Sanjit Dandapanthula and Aaditya Ramdas. Offline changepoint localization using a matrix of conformal p-values. *arXiv preprint arXiv:2505.00292*, 2025.

Alexander Philip Dawid and Allan M Skene. 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.

Guneet S. Dhillon, Javier Gonzalez, Teodora Pandeva, and Alicia Curth. E-scores for (in)correctness assessment of generative model outputs. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.

Senay A Gebreab, Khaled Salah, Raja Jayaraman, Muhammad Habib ur Rehman, and Samer Ellaham. LLM-based framework for administrative task automation in healthcare. In *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–7. IEEE, April 2024.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Peter Grünwald, Rianne de Heide, and Wouter M Koolen. Safe testing. In *2020 Information Theory and Applications Workshop (ITA)*, pages 1–54. IEEE, 2020.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

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

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Kexin Huang, Ying Jin, Ryan Li, Michael Y Li, Emmanuel Candes, and Jure Leskovec. Automated hypothesis validation with agentic sequential falsifications. *Forty-second International Conference on Machine Learning*, 2025a.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, et al. Biomni: A general-purpose biomedical AI agent. *biorxiv*, 2025b.

Salim I Amoukou, Tom Bewley, Saumitra Mishra, Freddy Lecue, Daniele Magazzeni, and Manuela Veloso. Sequential harmful shift detection without labels. *Advances in Neural Information Processing Systems*, 37:129279–129302, 2024.

Sooyong Jang, Sangdon Park, Insup Lee, and Osbert Bastani. Sequential covariate shift detection using classifier two-sample tests. In *International Conference on Machine Learning*, pages 9845–9880. PMLR, 2022.

Disi Ji, Padhraic Smyth, and Mark Steyvers. Can I trust my fairness metric? Assessing fairness with unlabeled data and Bayesian inference. *Advances in Neural Information Processing Systems*, 33: 18600–18612, 2020.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Ramesh Johari, Pete Koomen, Leonid Pekelis, and David Walsh. 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*, pages 1517–1525, 2017.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Shi Xuan Leong, Caleb E Griesbach, Rui Zhang, Kourosh Darvish, Yuchi Zhao, Abhijoy Mandal, Yunheng Zou, Han Hao, Varinia Bernales, and Alán Aspuru-Guzik. Steering towards safe self-driving laboratories. *Nat. Rev. Chem.*, 9(10):707–722, October 2025.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, 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*, pages 2757–2791, 2025.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017.

Gary Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, pages 1897–1908, 1971.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. OctoTools: An agentic framework with extensible tools for complex reasoning. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stanczak, Peter Shaw, Christopher Pal, and Siva Reddy. AgentRewardBench: Evaluating automatic evaluations of web agent trajectories. In *Second Conference on Language Modeling*, 2025.

Christopher Mohri and Tatsunori Hashimoto. Language models with conformal factuality guarantees. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Siddharth Narayanan, James D. Braza, Ryan-Rhys Griffiths, Manvitha Ponnapati, Albert Bou, Jon M Laurent, Ori Kabeli, Geemi Wellawatte, Sam Cox, Samuel G Rodriques, and Andrew White. Aviary: Training language agents on challenging scientific tasks. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025.

Young-Jin Park, Kristjan Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. Know what you don't know: Uncertainty calibration of process reward models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

Aleksandr Podkopaev and Aaditya Ramdas. Tracking the risk of a deployed model and detecting harmful distribution shifts. In *International Conference on Learning Representations*, 2022.

Drew Prinster, Xing Han, Anqi Liu, and Suchi Saria. WATCH: Adaptive monitoring for AI deployments via weighted-conformal martingales. In *Forty-second International Conference on Machine Learning*, 2025.

Yuanhao Qu, Kaixuan Huang, Ming Yin, Kanghong Zhan, Dyllan Liu, Di Yin, Henry C Cousins, William A Johnson, Xiaotong Wang, Mihir Shah, Russ B Altman, Denny Zhou, Mengdi Wang, and Le Cong. CRISPR-GPT for agentic automation of gene-editing experiments. *Nat. Biomed. Eng.*, pages 1–14, 2025.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling. In *The Twelfth International Conference on Learning Representations*, 2024.

Aaditya Ramdas and Tudor Manole. Randomized and exchangeable improvements of Markov's, Chebyshev's and Chernoff's inequalities. *Statistical Science*, 41(1):121–142, 2026.

Aaditya Ramdas and Ruodu Wang. Hypothesis testing with e-values. *Foundations and Trends in Statistics*, 1(1-2):1–390, 2025.

Mona Schirmer, Metod Jazbec, Christian A. Naesseth, and Eric Nalisnick. Monitoring risks in test-time adaptation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

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

Divya M Shanmugam, Shuvom Sadhuka, Manish Raghavan, John Guttag, Bonnie Berger, and Emma Pierson. Evaluating multiple models using labeled and unlabeled data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature*, pages 1–3, 2025.

Alexander Timans, Rajeev Verma, Eric Nalisnick, and Christian A. Naesseth. On continuous monitoring of risk violations under unknown shift. In Silvia Chiappa and Sara Magliacane, editors, *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pages 4204–4226. PMLR, 2025.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Hemish Veeraboina. AIME problem set 1983-2024, 2023. URL `https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024`.

Jean Ville. *Etude critique de la notion de collectif*, volume 3. Gauthier-Villars Paris, 1939.

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

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

Vladimir Vovk, Ivan Petej, Ilia Nouretdinov, Ernst Ahlberg, Lars Carlsson, and Alex Gammerman. Retrain or not retrain: Conformal test martingales for change-point detection. In *Conformal and Probabilistic Prediction and Applications*, pages 191–210. PMLR, 2021.

Abraham Wald and Jacob Wolfowitz. Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics*, pages 326–339, 1948.

Hanchen Wang, Yichun He, Paula P Coelho, Matthew Bucci, Abbas Nazir, Bob Chen, Linh Trinh, Serena Zhang, Kexin Huang, Vineethkrishna Chandrasekar, et al. SpatialAgent: An autonomous AI agent for spatial biology. *bioRxiv*, pages 2025–04, 2025.

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

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 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)*, pages 9426–9439, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024b.

Ian Waudby-Smith, Edward H Kennedy, and Aaditya Ramdas. Distribution-uniform anytime-valid sequential inference. *arXiv preprint arXiv:2311.03343*, 2023.

Ian Waudby-Smith, Lili Wu, Aaditya Ramdas, Nikos Karampatziakis, and Paul Mineiro. Anytime-valid off-policy inference for contextual bandits. *ACM/IMS Journal of Data Science*, 1(3):1–42, 2024.

Peter Welinder, Max Welling, and Pietro Perona. A lazy man's approach to benchmarking: Semisupervised classifier evaluation and recalibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3262–3269, 2013.

Menghua Wu, Cai Zhou, Stephen Bates, and Tommi Jaakkola. Thought calibration: Efficient and confident test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14291–14305. Association for Computational Linguistics, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380. Association for Computational Linguistics, 2018.

Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. Benchmarking LLMs via uncertainty quantification. *Advances in Neural Information Processing Systems*, 37:15356–15385, 2024.

Weiqiu You, Anton Xue, Shreya Havaldar, Delip Rao, Helen Jin, Chris Callison-Burch, and Eric Wong. Probabilistic soundness guarantees in LLM reasoning chains. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7517–7536, 2025.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024. Association for Computational Linguistics, 2025.

A THEORY

A.1 PROOF OF PROPOSITION 2.1

Proposition 2.1 (PAC thresholding). 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,

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

Proof. Suppose our calibration set contains n successful trajectories that are i.i.d. according to the null. Note that $\Pr_{\mathcal{H}_N}(\nexists t : M_t > c_\alpha | \mathcal{D}_{\text{cal}}) = \Pr_{\mathcal{H}_N}(\max_{t\in\mathbb{N}} M_t \leq c_\alpha | \mathcal{D}_{\text{cal}})$, 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

$$\Pr_{\mathcal{D}_{\text{cal}}}(c_\alpha < q_{1-\alpha}) \leq \delta. \tag{5}$$

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

$$\Pr_{\mathcal{H}_N}\left(\max_{t\in\mathbb{N}} M_t > c_\alpha \mid \mathcal{D}_{\text{cal}}\right) \leq \Pr_{\mathcal{H}_N}\left(\max_{t\in\mathbb{N}} M_t > q_{1-\alpha} \;\middle|\; \mathcal{D}_{\text{cal}}\right) = 1 - F(q_{1-\alpha}) \leq 1-(1-\alpha) = \alpha. \tag{6}$$

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

$$\Pr_{\mathcal{D}_{\text{cal}}}\left(\Pr_{\mathcal{H}_N}\left(\exists t : M_t > c_\alpha | \mathcal{D}_{\text{cal}}\right) \leq \alpha\right) = \Pr_{\mathcal{D}_{\text{cal}}}\left(\Pr_{\mathcal{H}_N}\left(\max_{t\in\mathbb{N}} M_t > c_\alpha \;\middle|\; \mathcal{D}_{\text{cal}}\right) \leq \alpha\right) \geq 1-\delta \tag{7}$$

as desired.

We construct a $(1-\delta)$-confidence upper bound on $q_{1-\alpha}$ using the following argument, which follows ideas from Clopper and 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

$$\Pr_{\mathcal{D}_{\text{cal}}}(M_{(k)} < x) = \Pr_{\mathcal{D}_{\text{cal}}}(K(x) \geq k) = \Pr(\text{Binomial}(n, F(x-)) \geq k). \tag{8}$$

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

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

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. equation 5 holds. That is, by construction, c_α satisfies

$$\Pr_{\mathcal{D}_{\text{cal}}}(c_\alpha < q_{1-\alpha}) = \Pr_{\mathcal{D}_{\text{cal}}}(M_{(k^\star)} < q_{1-\alpha}) \leq \Pr(\text{Binomial}(n, 1-\alpha) \geq k^\star) \leq \delta, \tag{10}$$

as desired.

□

Proposition A.1 (Log-optimality of density-ratio statistic)**.** *Let p_0 and p_1 be the alternative and null densities, respectively. Introduce an absorbing terminal symbol † and extend the process by setting $S_t = \dagger$ for every $t > T$. Thus, $S_{T+1} = \dagger$ records termination and all later observations are also †. Define*

$$M_0 = 1, \qquad M_t = \frac{p_0(\mathbf{S}_{[1:t]})}{p_1(\mathbf{S}_{[1:t]})}.$$

Assume $P_0^{(t)} \ll P_1^{(t)}$ for every t, where $P_0^{(t)}$ and $P_1^{(t)}$ denote the distributions of $\mathbf{S}_{[1:t]}$ under the alternative and null, respectively.

Then $(M_t)_{t=0}^{\infty}$ is a test martingale under $\mathcal{H}_N$ and hence an e-process. Moreover, for any other e-process $(M'_t)_{t=0}^{\infty}$ and any stopping time τ for which the expectations are well-defined,

$$\mathbb{E}_{\mathcal{H}_A}[\log M_\tau] \geq \mathbb{E}_{\mathcal{H}_A}[\log M'_\tau].$$

Proof. Let $\mathcal{F}_t = \sigma(\mathbf{S}_{[1:t]})$ and let L be a reference measure on $\overline{\mathcal{S}} = \mathcal{S} \cup \{\dagger\}$ with an atom at †. Density ratios are nonnegative and $M_0 = 1$. Under the null, $\mathbf{S} \sim P_1$, and

$$\mathbb{E}_{\mathcal{H}_N}\left[M_t \mid \mathcal{F}_{t-1}\right] = \int_{\overline{\mathcal{S}}} \frac{p_0(\mathbf{S}_{[1:t-1]}, s_t)}{p_1(\mathbf{S}_{[1:t-1]}, s_t)} p_1(s_t \mid \mathbf{S}_{[1:t-1]})\, L(ds_t).$$

That equals:

$$\begin{aligned}
&\frac{p_0(\mathbf{S}_{[1:t-1]})}{p_1(\mathbf{S}_{[1:t-1]})} \int_{\overline{\mathcal{S}}} \frac{p_0(s_t \mid \mathbf{S}_{[1:t-1]})}{p_1(s_t \mid \mathbf{S}_{[1:t-1]})} p_1(s_t \mid \mathbf{S}_{[1:t-1]})\, L(ds_t) \\
&\quad = M_{t-1} \int_{\overline{\mathcal{S}}} p_0(s_t \mid \mathbf{S}_{[1:t-1]})\, L(ds_t) \\
&\quad = M_{t-1}.
\end{aligned}$$

The integral includes both another verifier score and the terminal symbol. Thus, $(M_t)_{t=0}^{\infty}$ is a test martingale under $\mathcal{H}_N$.

Equivalently, define the posterior odds

$$O_t := \frac{\Pr(Y = 0 \mid \mathbf{S}_{[1:t]})}{\Pr(Y = 1 \mid \mathbf{S}_{[1:t]})}.$$

Bayes' rule gives

$$O_t = \frac{p_0(\mathbf{S}_{[1:t]}) \Pr(Y = 0)}{p_1(\mathbf{S}_{[1:t]}) \Pr(Y = 1)},$$

and therefore

$$\begin{aligned}
M_t &= \frac{O_t}{O_0} \\
&= \prod_{i=1}^{t} \frac{O_i}{O_{i-1}} \\
&= \prod_{i=1}^{t} \frac{p_0(S_i \mid \mathbf{S}_{[1:i-1]})}{p_1(S_i \mid \mathbf{S}_{[1:i-1]})}.
\end{aligned}$$

Once $S_{T+1} = \dagger$, no further information arrives, so $O_i = O_{T+1}$ for every $i \geq T + 1$ and all later increments equal one.

We now establish log-optimality. In the non-sequential setting, the likelihood ratio

$$E = \frac{p_0}{p_1}$$

is the log-optimal e-variable. It suffices to consider

$$E' = \frac{dQ}{dP_1} = \frac{q}{p_1}$$

for distributions Q satisfying $Q \ll P_0 \ll P_1$. Then

$$\begin{aligned}\mathbb{E}_{\mathcal{H}_A}\left[\log \frac{E'}{E}\right] &= \int p_0(x) \log\left(\frac{q(x)/p_1(x)}{p_0(x)/p_1(x)}\right) L(dx) \\ &= -\int p_0(x) \log\left(\frac{p_0(x)}{q(x)}\right) L(dx) \\ &= -D_{\mathrm{KL}}(P_0 \| Q) \\ &\leq 0.\end{aligned}$$

Therefore,

$$\mathbb{E}_{\mathcal{H}_A}[\log E'] \leq \mathbb{E}_{\mathcal{H}_A}[\log E].$$

The sequential extension to e-processes and stopping times is Theorem 7.11 of Ramdas and Wang (2025), which gives

$$\mathbb{E}_{\mathcal{H}_A}[\log M'_\tau] \leq \mathbb{E}_{\mathcal{H}_A}[\log M_\tau].$$

This justifies the use of the density ratio process as a target estimand. □

A.2 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.3 BASELINE: RANDOMIZED VILLE

Ramdas and 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.2. *(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 and 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.4 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.

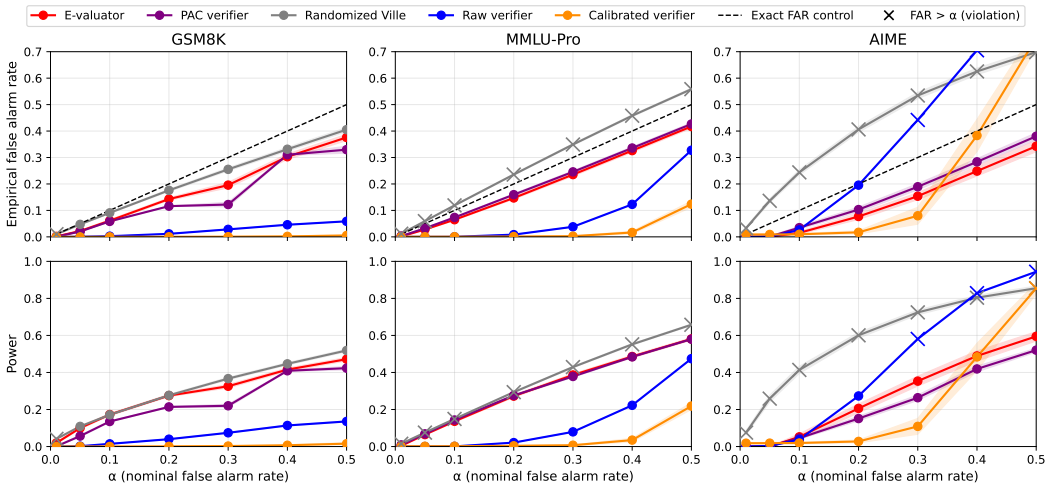

Figure 4: **GSM8k, MMLU-Pro, and AIME results**. The false alarm rate is empirically controlled for *e-valuator* and the PAC verifier. Additionally, *e-valuator* achieves greater (or equal) power among methods that are able to control the false alarm rate.

B Additional Results

B.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 3.1): GSM8k (Cobbe et al., 2021), MMLU-Pro (Wang et al., 2024b), and AIME (Veeraboina, 2023) (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.

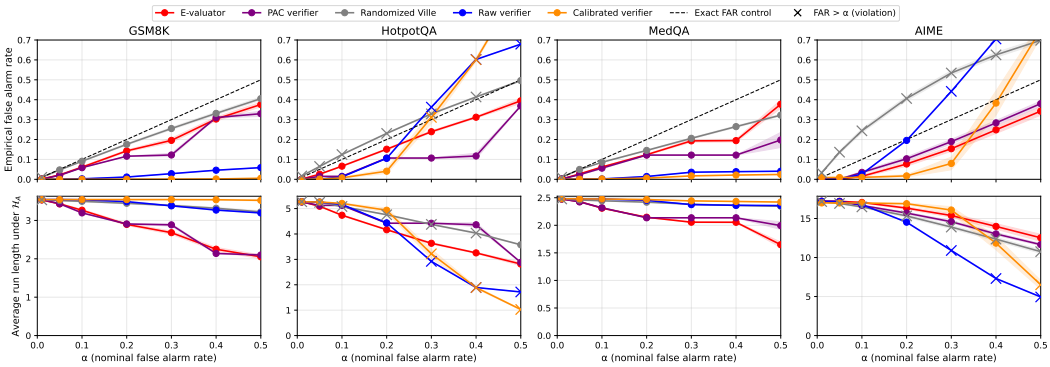

Figure 5: **Average run length results**. The average run length for unsuccessful trajectories is generally lower for e-valuator than other methods, indicating that it is able to detect unsuccessful trajectories earlier. Violations of FAR control are marked with an X.

B.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 (Veeraboina, 2023) (Figure 5). E-valuator provides better average run length than the PAC verifier in all but one dataset (AIME), on which it has favorable power (Figure 4). 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.

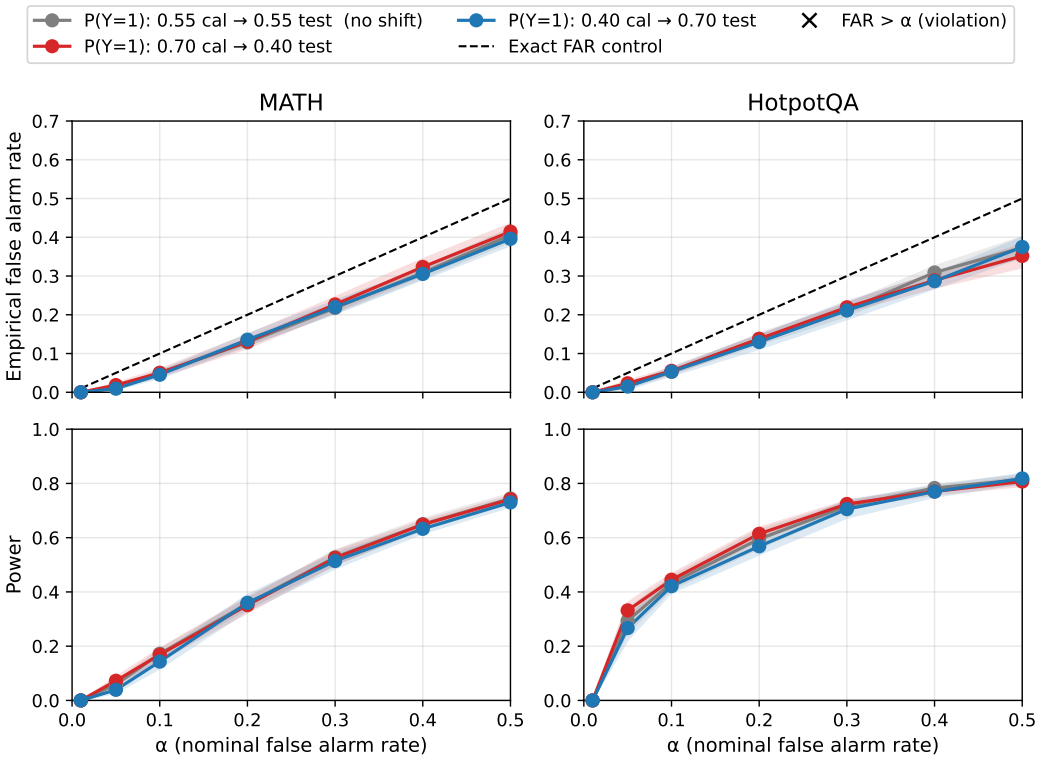

Figure 6: **Label shift results**. On both MATH and HotpotQA, shifting the test split to have a different proportion of $Y = 1$ than the calibration split does not impact the power or FAR observed.

B.3 Robustness to distribution shift

B.3.1 Label shift

We first note that in the case of label shift alone, i.e., cases where $p(Y = 1)$ changes between calibration and test but $p(\mathbf{S}|Y)$ changes, our PAC procedure from Algorithm 1 will remain valid. To see this, note that our null samples are all drawn from cases where $Y = 1$. Label shift indicates that the relative proportions of null/alternative are changing, but the procedure will still be calibrated if the null itself doesn't shift. Similarly, if $p(\mathbf{S}|Y = 1)$ doesn't shift, our power will be constant between calibration and test as well.

To empirically test this guarantee, we induced label shift by sampling different proportions of $Y = 1$ in the calibration and test split. Figure 6 shows these results; our method is robust to label shift, as predicted by theory.

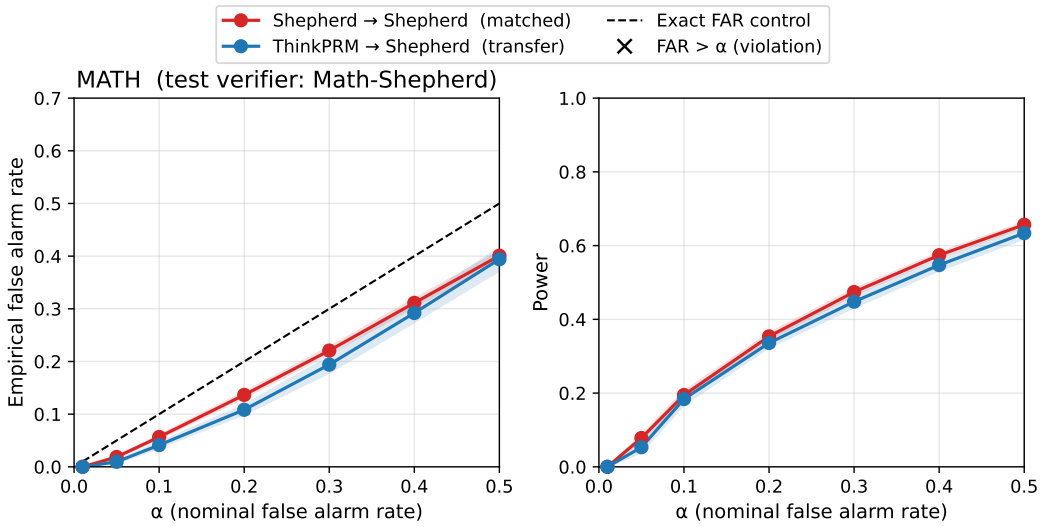

Figure 7: **Verifier shift results**. We study shifts in the verifier used between calibration and test. In particular, we see that using the same verifier in both calibration and test has superior power (but only mildly) compared to switching the verifier between test and training.

B.3.2 Verifier shift

Next we compare other types of (potentially unknown) shifts. In general, we cannot provide guarantees for unknown and arbitrary shifts. However, in this section we empirically change one component (the verifier) while keeping the dataset and agent fixed.

We learn the density ratios and thresholds using one verifier than transfer them as-is to another. Here, we compared using Math-Shepherd in both calibration and test to using ThinkPRM Khalifa et al. (2025) in calibration and Math-Shepherd (Wang et al., 2024a) during test (Figure 7). E-valuator maintains strong power even when swapping the verifier.

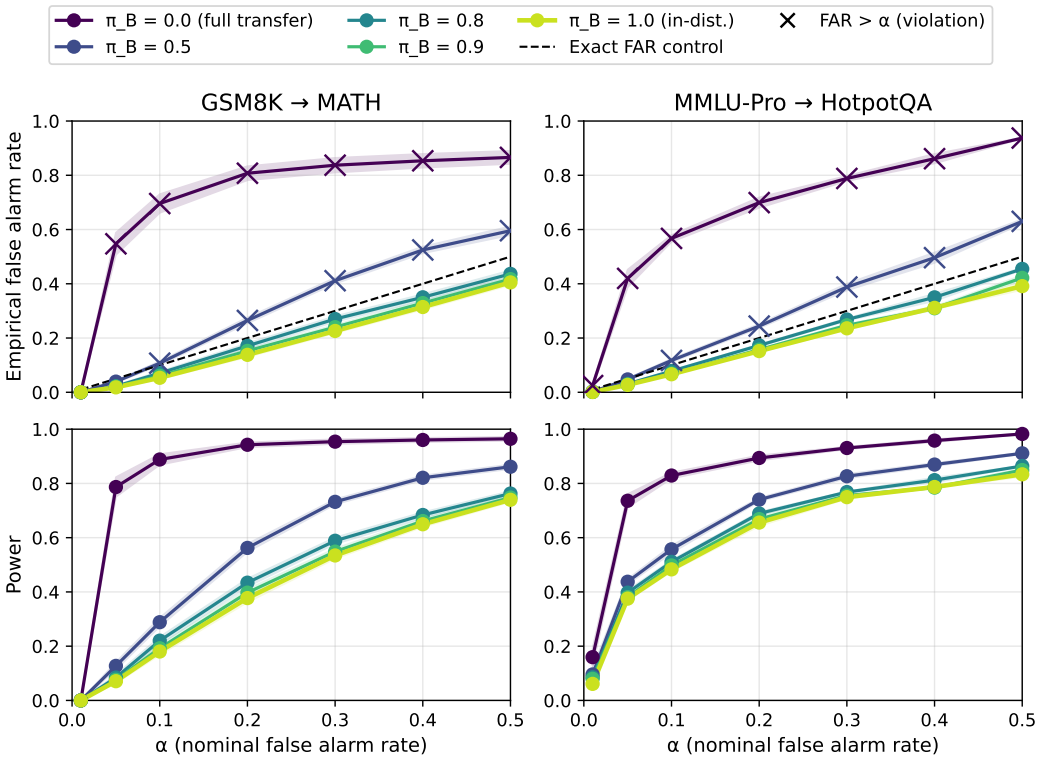

Figure 8: **Dataset shift results**. We study shifts beween GSM8k and MATH (left) and MMLU-Pro and HotpotQA (right) by mixing in different proportions of the test dataset into the calibration one. Full transfer does not provide FAR guarantees. However, once about 80% of the calibration set is from the test set, e-valuator is able to empirically control FAR.

B.3.3 Dataset shift

Finally, we study the robustness of e-valuator to dataset shifts. We empirically tested distribution shifts by learning both the threshold and the density ratios on one dataset ("dataset A") and using them as-is on another ("dataset B"). To study how the magnitude of distribution shifts between the calibration and test data affect error rates, we mixed dataset B in different proportions into the calibration split. We keep the learned thresholds and density ratios constant between calibration and test.

In particular we mix the test dataset (dataset B) in different fractions (π_B) into the calibration set (Figure 8). E-valuator is able to control FAR once a majority of (but not all) the calibration set is from the test distribution, indicating some empirical robustness to distribution shift.

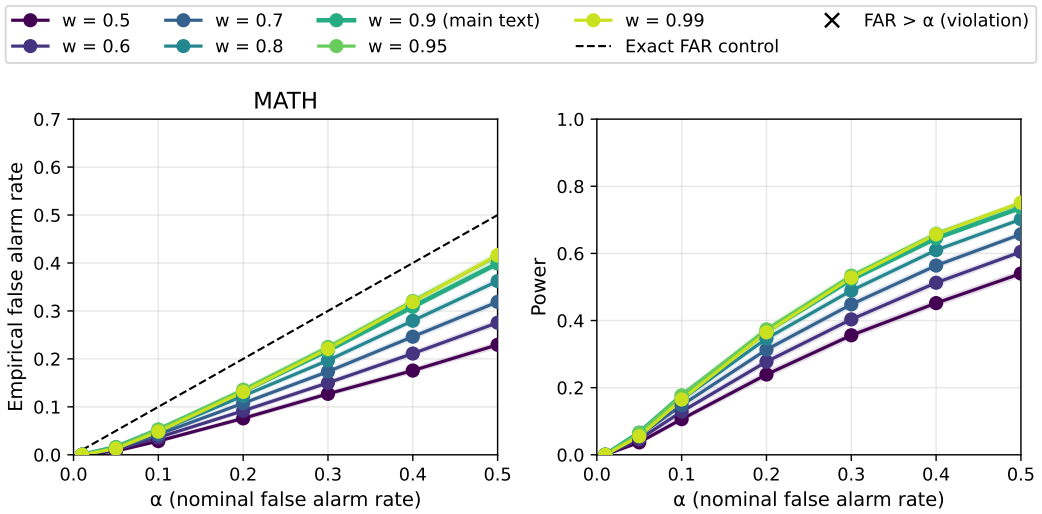

Figure 9: α', δ. We set $\alpha' = w \cdot \alpha$ and $\delta = (1 - w) \cdot \alpha$ and ablate w. Larger values of w provide higher power, as long as there are enough trajectories to accomodate that w.

B.4 Ablations of α', δ split

In the main text, we control the FAR at level α by choosing α', δ such that $\alpha' + \delta = \alpha$ in Algorithm 1. Specifically, we set $\alpha' = w \cdot \alpha$ and $\delta = (1 - w) \cdot \alpha$ and $w = 0.9$. Here we ablate w from 0.5 to 0.99 on the MATH dataset, keeping the calibration set size fixed at 1000 trajectories.

Doing so, we observe that larger values of w increase power. It is worth noting that one cannot set w too large; as there may not be enough calibration set trajectories to accomodate the guarantee. As we note in the main section, the choice of α' and δ is constrained by the number of samples with $Y = 1$ the threshold calibration set, denoted n. If n is too small, the algorithm returns $c_\alpha = \infty$ and the procedure never rejects (trivially controlling FAR but obtaining zero power). This occurs when $n < \log \delta / \log(1 - \alpha')$. Thus, we recommend setting α', δ such that $n \geq \lceil \log \delta / \log(1 - \alpha') \rceil$.

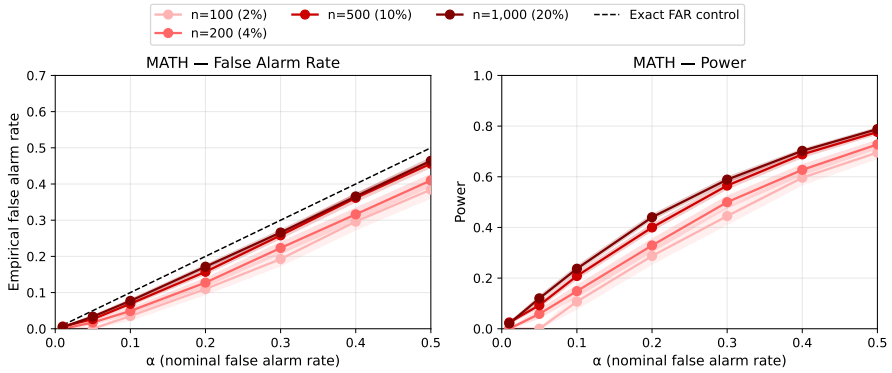

Figure 10: **Calibration set size**. On the MATH dataset, the false alarm rate is empirically controlled for all calibration set sizes. E-valuator tends to be more conservative for very small n and α, as expected.

B.5 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 10, 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).

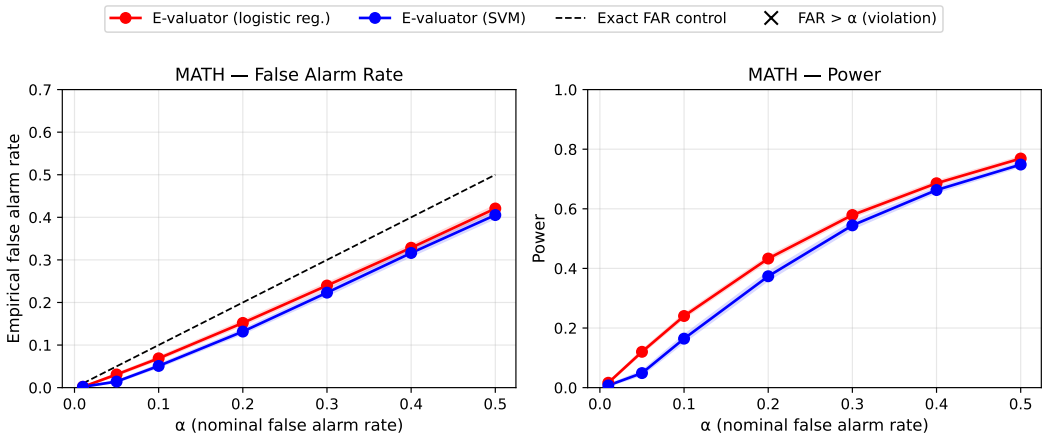

Figure 11: **Density ratio estimator**. Changing the density ratio estimator from logistic regression to SVM does not substantially change the results on the MATH dataset. False alarm rate control and power track closely across both density ratio estimation methods.

B.6 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 11).

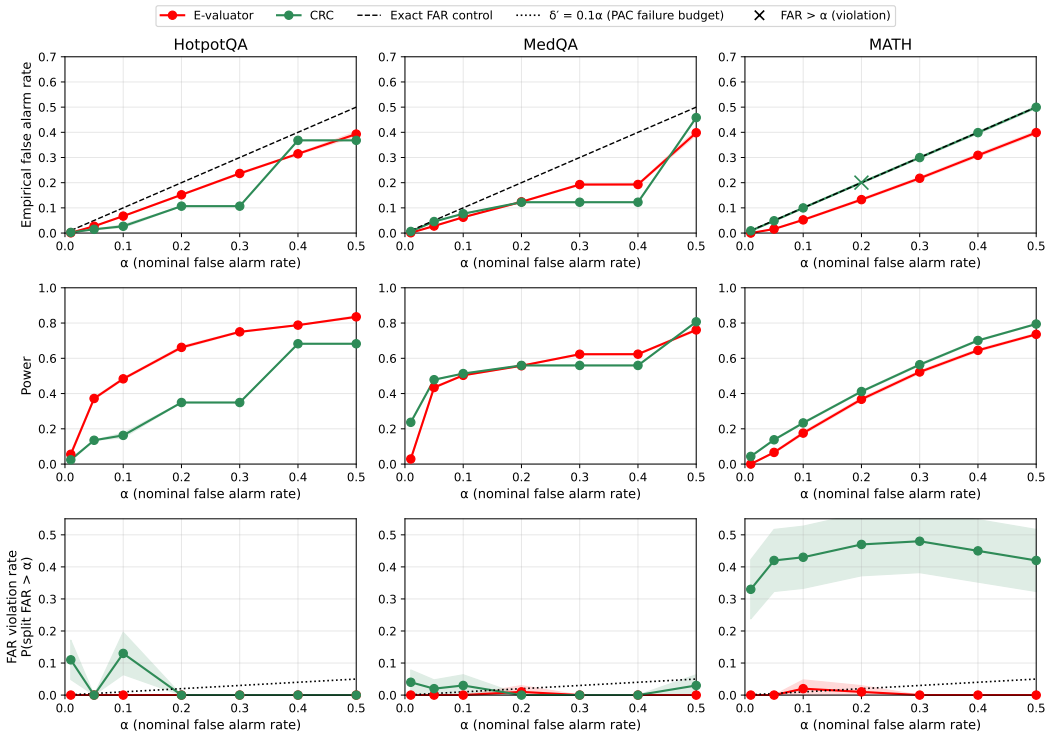

Figure 12: **Comparison to conformal risk control (CRC)**. We compare e-valuator to conformal risk control on the raw verifier scores. This baseline sometimes achieves better marginal FAR control and power (right). However, because CRC only controls FAR in expectation, it can often inflate the FAR violation rate (over different calibration splits) drastically (right).

B.7 Conformal risk control

In practice, one draws a single calibration set $\mathcal{D}_{\text{cal}}$, computes a single threshold c_α, and deploys it. The quantity that matters is therefore the *realized* false alarm rate $\text{FAR}(c_\alpha)$ of that particular threshold, which is a random variable through $\mathcal{D}_{\text{cal}}$. Our PAC thresholding scheme from Proposition 2.1 (used by both e-valuator and the PAC verifier baseline) controls this quantity with high probability: with probability at least $1-\delta$ over the draw of $\mathcal{D}_{\text{cal}}$, $\text{FAR}(c_\alpha) \leq \alpha'$. With our split $\alpha' + \delta = \alpha$, this means that on at least a $1-\delta$ fraction of calibration splits, the realized FAR is at most $\alpha' < \alpha$, and by the decomposition of the marginal false alarm rate in the main text, the marginal FAR is at most $\alpha' + \delta = \alpha$. Note that this does *not* rule out violations: on the remaining (at most) δ fraction of "unlucky" splits, the realized FAR can exceed α', and indeed exceed α. What the PAC guarantee provides is a bound on how *often* this happens.

Conformal risk control (Angelopoulos et al., 2024) instead targets the marginal quantity directly, choosing c_α so that

$$\mathbb{E}_{\mathcal{D}_{\text{cal}}}\left[\text{FAR}(c_\alpha)\right] = \mathbb{E}_{\mathcal{D}_{\text{cal}},\, M_{1:T}}\left[\mathbb{1}(\exists\, t \in [T] : M_t > c_\alpha)\right] \leq \alpha. \tag{12}$$

This bounds the FAR only on average over calibration sets and says nothing about the realized FAR of the threshold one actually deploys. A bound on the mean is compatible with frequent violations: for instance, a procedure whose realized FAR is $\alpha/2$ on half of the calibration splits and $3\alpha/2$ on the other half satisfies Eq. 12 while exceeding α half of the time. Conformal risk control therefore provides no bound on the fraction of splits on which the realized FAR exceeds α, whereas PAC thresholding bounds this fraction by δ. The price of the PAC guarantee is that the nominal level is split between α' and δ, which reduces power relative to spending the full budget α on the threshold.

We compare to this baseline and empirically compare the rate of false alarm rate violation, i.e., the fraction of random calibration splits on which the realized FAR exceeds α. For e-valuator and the

PAC verifier, this fraction is at most δ by Proposition 2.1; conformal risk control provides no such bound. We thus compare to the more directly comparable baseline (PAC verifier) in the main and to conformal risk control here.

As shown in Figure 12, CRC sometimes achieves comparable or better marginal FAR control, at the cost of violating the FAR bound on many calibration splits. For instance, on the MATH dataset, even though CRC obtains better power and more aggressive FAR, it violates FAR on more than 40% of the calibration splits tested. In comparison, e-valuator always keeps the FAR low across calibration splits.

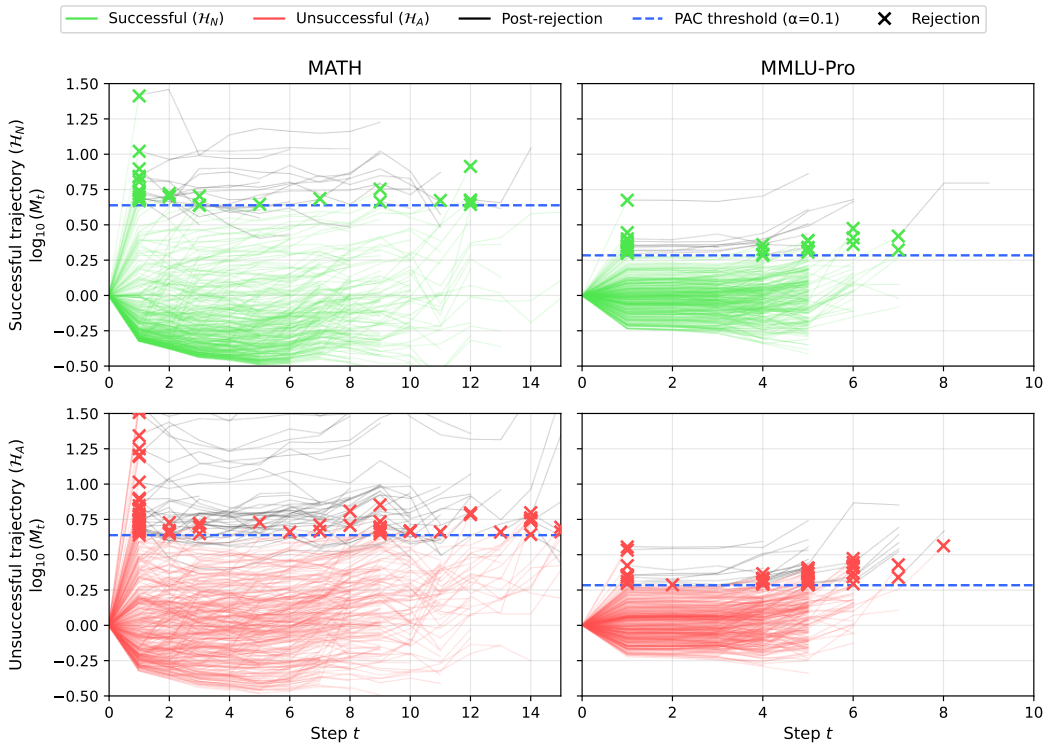

Figure 13: **Example sequences**. On the MATH dataset (left column), many of the rejections are generated at M_1, after the first action, indicating the first action of the agent is important in deciding success. On the MMLU-Pro dataset (right column), which has shorter trajectories, there are more rejections later in the sequence.

B.8 Example M_t sequences

We additionally show some example M_t sequences (plotted as $\log(M_t)$ for visual clarity) in Figure 13 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.

C 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 C.

For **mathematical reasoning**, we use GSM8k (Cobbe et al., 2021) for our tool-calling agent experiments and MATH (Hendrycks et al., 2021) and AIME (Veeraboina, 2023) 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 C lists all the datasets, agents, and verifiers used. Table 2 additional lists some statistics on the quality of these verifiers, and Table 3 lists the size of each dataset. 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., 2025)	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 (Veeraboina, 2023)	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.

Dataset	Total	Successful (null)	Unsuccessful	Nulls in $\mathcal{D}_{\text{threshold}}$
AIME	911	549 (60.3%)	362	54.8
GSM8k	5,978	5,012 (83.8%)	966	499.2
HotpotQA	16,110	7,799 (48.4%)	8,311	773.9
MATH	5,000	2,862 (57.2%)	2,138	289.8
MMLU-Pro	12,032	7,532 (62.6%)	4,500	750.3
MedQA	10,178	8,224 (80.8%)	1,954	823.6

Table 3: Dataset sizes. For each dataset, we report the total number of trajectories, the number of successful ($Y = 1$, null) and unsuccessful ($Y = 0$) trajectories, and the number of successful trajectories in $\mathcal{D}_{\text{threshold}}$ (i.e., the number of samples available to Algorithm 1), averaged over the 50 random calibration splits used in the main text.

C.1 Additional computational details

We use the default hyperparameter settings in `scikit-learn` with logistic regression for all experiments presented in this paper. We split $\mathcal{D}_{cal}$ 50/50 into the density ratio estimation and threshold calibration split. Given a set of verifier scores, all experiments in this paper can be completed in under a minute on a standard laptop.

D ADDTIONAL RELATED WORK

Verifiers and PRMs. As *e-valuator* is a statistical wrapper for any verifier, our work is relevant to prior work on building better verifiers. These verifiers are often trained as models that estimate a reward (e.g., correctness) 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 not (Uesato et al., 2022; Wang et al., 2024a; Khalifa et al., 2025). Some prior works also calibrate existing PRMs (You et al., 2025; Park et al., 2026), although calibration alone is insufficient to control false alarm rates. 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 include LLM-as-a-judge (Bavaresco et al., 2025) and outcome reward models, which only provide a label for the entire trajectory (Creswell et al., 2022). There are also several benchmarks to compare verifiers (Lù et al., 2025; Zheng et al., 2025).

Sequential hypothesis testing and e-values. E-valuator directly builds on prior work in e-values, which have useful properties for sequential hypothesis testing (Ramdas and Wang, 2025; Vovk and Wang, 2021; 2023). E-values provide anytime validity over a (potentially infinite) sequence of tests (Wang and Ramdas, 2025; Waudby-Smith et al., 2024; 2023; Grünwald et al., 2020). They have found applications in many settings, including A/B testing (Johari et al., 2017), changepoint detection (Shin et al., 2023; Lorden, 1971), and others (Chen and Wang; Huang et al., 2025a; Dhillon et al., 2026).

Hypothesis testing for AI monitoring. Some prior works build hypothesis tests to monitor AI deployments, albeit in different contexts. Vovk et al. (2021) propose using conformal test martingales (CTMs) for continual monitoring of AI deployments, and Prinster et al. (2025) develop weighted CTMs to enable test-time adaptation and analyze the cause of degradation. Safe anytime-valid testing has also been applied to track the risk of deployed models Podkopaev and Ramdas (2022) and sequentially test if a classifier is fair (Chugg et al., 2023), among other applications (I Amoukou et al., 2024; Schirmer et al., 2026; Timans et al., 2025). (Jang et al., 2022) applies classifier two-sample tests (Lopez-Paz and Oquab, 2017) to detect covariate shifts. More broadly, several works model class-conditional densities of classifier scores and the corresponding density ratios for evaluation (Shanmugam et al., 2025; Welinder et al., 2013; Dawid and Skene, 1979; Ji et al., 2020).

Statistical guarantees on LLM outputs. Another line of work tries to add formal statistical controls to LLMs. Conformal prediction methods (Shafer and 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., 2024). These ideas have been further applied to control the factuality of LLM outputs (Cherian et al., 2024; Mohri and 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.