Judge Drift Auditing: Operating Characteristics of a Prediction-Powered e-Detector for LLM-Judge Drift under a Fixed Gold Budget

Andrew Sheng-Han Huang*
National Taiwan University
johnshhuang1111@gmail.com

Abstract

The LLM-as-judge paradigm underpins modern RLHF, leaderboards, and product evaluation, but a judge is itself a model behind an API: a silent version bump or a scoring-prompt edit can shift how it scores, and practitioners routinely complain that "our judge drifted." A monitor for this should combine the abundant cheap judge labels with a sparse, expensive gold standard, and should control false alarms even though it is watched continuously. The recent, concurrent work of Li [8] supplies exactly such a monitor — a prediction-powered e-process with an anytime-valid guarantee — and frames it as *attribution* (is the system or the judge to blame?). What that work, by its own statement, does not provide is the monitor's *operating characteristic*: how detection delay depends on the drift magnitude and on the gold-label budget, and how it compares against the change-detection baselines a practitioner would otherwise reach for. We supply this characterization, and our central, deliberately unflattering finding is that **prediction-powered sequential detection does not dominate simple fixed-n testing for judge drift under realistic budgets.** We build a prediction-powered, weighted-start e-detector (a Shin–Ramdas–Rinaldo mixture of betting supermartingales over the rectified judge-versus-gold residual) whose false-alarm *probability* under indefinite monitoring is at most α by Ville's inequality, and we measure: (i) in matched null simulations (3000 streams, horizon 1500), an empirical false-alarm rate of 0.00% at nominal 5%; (ii) an operating-characteristic curve of detection delay versus drift magnitude, well summarized by the single variable $\rho\Delta^2$ ($R^2 = 0.97$ across the combined magnitude-and-budget sweep) but with a *measured, sub-linear* log–log slope of 0.58 (95% CI $[0.51, 0.66]$), so we report an empirical monotone scaling, not a clean closed-form law; (iii) a gold-budget frontier showing delay falling roughly as $1/\rho$ in the gold rate, with the prediction-powered rectifier detecting a 0.12-magnitude drift a modest 1.01× faster than a *true* gold-only detector (one that uses no cheap proxy) at the same gold rate — the prediction-powering buys a small but real gain; and (iv) a head-to-head against periodic fixed-n re-testing and an offline PELT oracle at matched gold budget and false-alarm level. There the result splits, and we report it honestly: our streaming detector *beats* the offline PELT oracle on reliable detection (100% vs. 6% at Δ=0.20), but *loses to periodic fixed-n on median detection delay at every drift size we test* (e.g. 468 vs. 180 items at Δ=0.20), because fixed-n spends its whole gold budget at one pre-set look and pays nothing for anytime validity. The honest contribution is thus a *characterization of the tradeoff* — the streaming detector buys anytime validity, unlimited monitoring, and schedule-freedom at a cost in median delay — which is what a practitioner needs to decide whether that trade is worth it for auditing a judge. We do *not* claim a new monitoring method (it is concurrent prior work's), and we do *not* claim it is faster; we claim the missing operating-characteristic measurement and a cautionary tradeoff that defers to the concurrent monitor.

*Code, all simulation seeds, and end-to-end reproduction scripts are available from the author. The statistical core is pure Python standard library; every number in this paper is generated by a script into `results/*.json` and injected as a LaTeX macro.

1 Introduction

A growing share of the field's evaluation runs through an *LLM judge*: a strong model is asked to score the outputs of other models, and those scores stand in for human preference in RLHF reward modeling, on chat-arena leaderboards, and in production quality monitoring [5, 15]. The arrangement is cheap and scalable, but it has a quietly load-bearing assumption: that the judge's scoring behavior is *stable*. It frequently is not. The judge is a model behind an API; a silent checkpoint update, a temperature change, or an edit to the scoring rubric can systematically shift how it grades, and the shift is invisible until someone notices the downstream numbers moved [8]. "Our judge drifted" is now a common complaint, and the question it raises is operational: *how would we know, how fast, and at what annotation cost?*

A principled answer has to respect three constraints at once. First, gold labels — the trustworthy human (or stronger-model) judgments against which drift is defined — are *expensive*, so a monitor must lean on the abundant cheap judge labels and spend gold sparingly. This is precisely the regime of prediction-powered inference (PPI), which debiases a cheap predictor with a small labeled sample [1, 2], recently extended to e-values [4] and packaged for evaluation workflows [10]. Second, the monitor is watched *continuously*: an operator looks at it whenever the dashboard updates and reacts when it crosses a line, so its error guarantee must hold at arbitrary, data-dependent stopping times — the domain of anytime-valid, game-theoretic statistics [6, 11, 14]. Third, drift is a *change over time*, so the right object is a sequential change detector, for which e-detectors provide a nonparametric, composite-null framework with clean false-alarm guarantees [12].

The concurrent state of the art, and the gap. Strikingly, a paper posted days before this one assembles exactly these three ingredients. Li [8] monitor an LLM judge with a prediction-powered e-process plus a betting e-process on a frozen human-anchor set, prove an anytime-valid false-alarm guarantee via Ville's inequality, and demonstrate detection on real judge changes. Their contribution is *attribution*: a guard-window rule that returns a verdict in $\{\text{none}, \text{system}, \text{judge}\}$, disentangling a worse product from a changed judge. We take their method as given and do not claim it. What their work explicitly does not provide — stated in their own limitations — is the monitor's *operating characteristic*: they report latency at a few fixed drift conditions, but "do not systematically sweep detection delay across a continuous range of drift magnitudes," and they "establish finite-sample, anytime-valid bounds …rather than asymptotic average-run-length formulas." Nor do they compare against the offline change-point detectors (e.g. PELT [7]) a practitioner might otherwise apply to a logged judge stream. Yet that operating characteristic is exactly what is needed to *provision* a monitor: how many gold labels per item to buy, and what detection delay to expect, for a target false-alarm level.

Contributions. This paper supplies the missing characterization — and the headline it produces is a negative one, which we believe is the more useful result. Concretely:

1. A prediction-powered, *weighted-start* e-detector for judge-versus-gold drift (Section 3): a Shin–Ramdas–Rinaldo mixture of betting supermartingales over a PPI-rectified residual, with summable start weights so that its false-alarm *probability* under indefinite monitoring is at most α by Ville's inequality applied to a dominating mixture that carries the not-yet-started future weight (Proposition 1). We are explicit that the running statistic is itself *not* a supermartingale and that the unweighted mixture controls only the average run length, not the ever-alarm probability — distinctions we verify empirically and that were self-caught bugs in our own first implementation.
2. An **operating-characteristic measurement** (Section 4): detection delay as a function of drift magnitude Δ at fixed gold rate. Delay is well summarized by the single variable $\rho\Delta^2$ ($R^2 = 0.97$), but the measured log–log slope is 0.58 (95% CI $[0.51, 0.66]$), sub-linear to the textbook first-order value of 1; we therefore report an *empirical monotone scaling*, not a clean closed-form expected-detection-delay law.
3. A **gold-budget frontier**: detection delay versus the gold-label rate ρ, and a measurement that the prediction-powered rectifier detects about $1.01\times$ faster than a *true* gold-only detector — one that ignores the cheap proxy entirely and updates only on gold-labeled items — at the same gold rate

when the cheap proxy is informative (with the kill-test that, if it did not, the "prediction-powered" qualifier would be empty).

4. A **head-to-head** against periodic fixed-n re-testing and offline PELT at a matched gold budget and matched false-alarm level (Section 4.1), reported as the honest split it is: the streaming detector attains a far higher detection rate than the offline PELT oracle on a sparse-gold-rectified stream, but **does not beat periodic fixed-n on median detection delay** — fixed-n is faster at every drift size we test. The deliverable is a *characterization of the tradeoff* (anytime validity and schedule-freedom vs. median delay), not a claim of superiority.
5. A **controlled live corroboration** with a single open-weight judge (Section 4.2): the judge re-scores a fixed, already-collected stream of answers and a known drift is induced mid-stream by a scoring-prompt change.

The statistical core is pure Python standard library; every number is produced by a script and injected as a macro, and re-running the analysis reproduces the JSON bit-for-bit.

2 Problem setup

A stream of items $t = 1, 2, \ldots$ arrives. For each, a cheap *judge* emits a score $s_t \in [0, 1]$ (e.g. a normalized quality rating, or a 0/1 acceptance). A trustworthy *gold* label $y_t \in [0, 1]$ exists but is observed only when a *gold mask* $m_t \in \{0, 1\}$ fires; the mask is drawn independently of the data with $\mathbb{E}[m_t \mid \mathcal{F}_{t-1}, s_t, y_t] = \rho$, so the average gold rate is a fixed $\rho \in (0, 1]$ (the *gold budget*). On every item a cheap *proxy* $\hat{y}_t \in [0, 1]$ for the gold is also available — for instance the judge's own thresholded score, or a small fixed predictor.

The monitored quantity is the judge-versus-gold *bias* $\mu_t = \mathbb{E}[s_t - y_t \mid \mathcal{F}_{t-1}]$, the systematic gap between what the judge says and what the gold says. Under the null H_0 of *no drift*, $\mu_t = \mu_0$ for all t, where μ_0 is a (possibly nonzero) stable baseline bias estimated before monitoring. *Drift* is an unknown onset $t^\star$ after which $\mu_t = \mu_0 + \Delta$ for a magnitude $\Delta \neq 0$ (a judge that became systematically more lenient, $\Delta > 0$ for $s - y$, or harsher, $\Delta < 0$). The goal is to raise an alarm as soon as possible after $t^\star$ while almost never alarming under H_0, spending gold at rate ρ.

The null is conditional. For the e-process construction below the relevant null is not a statement about marginal means but the *conditional* one-sided hypothesis on the rectified increment z_t (defined in (2)): writing the detector for upward drift,

$$H_0: \quad \mathbb{E}[\, z_t - \mu_0 \mid \mathcal{F}_{t-1}\,] \;\le\; 0 \quad \text{for all } t, \tag{1}$$

and symmetrically $\mathbb{E}[\, -(z_t - \mu_0) \mid \mathcal{F}_{t-1}\,] \le 0$ for the downward detector. This conditional, predictable-mean form (not merely a marginal-mean assumption) is what makes each one-step factor a genuine conditional e-value and licenses Ville's inequality; our i.i.d. Bernoulli gold mask, independent of the data given $\mathcal{F}_{t-1}$, satisfies it.

3 A prediction-powered weighted-start e-detector

The prediction-powered increment. The key object is a per-item signal that is cheap to compute, unbiased for the bias μ_t, and low-variance when the proxy is good:

$$z_t \;=\; \underbrace{(s_t - \hat{y}_t)}_{\text{available every item}} \;+\; \underbrace{\tfrac{m_t}{\rho}\,(\hat{y}_t - y_t)}_{\text{rectifier, only on gold items}} \quad . \tag{2}$$

Because m_t is independent of the data with mean ρ, $\mathbb{E}[z_t \mid \mathcal{F}_{t-1}] = \mathbb{E}[s_t - \hat{y}_t \mid \mathcal{F}_{t-1}] + \mathbb{E}[\hat{y}_t - y_t \mid \mathcal{F}_{t-1}] = \mathbb{E}[s_t - y_t \mid \mathcal{F}_{t-1}] = \mu_t$ for *any* proxy $\hat{y}$ — the PPI control-variate identity [1]. When $\hat{y}_t \approx y_t$, the always-available term $s_t - \hat{y}_t$ carries the signal and the ρ-reweighted correction is small, so z_t has lower variance than the gold-only signal $\frac{m_t}{\rho}(s_t - y_t)$; this variance reduction is what buys faster detection per gold label.

Centering and the betting increment. Map z_t into $[0,1]$ by $w_t = (z_t - \mu_0 + B)/(2B)$, where B is an almost-sure bound on $|z_t - \mu_0|$. Since $|s_t - \hat{y}_t| \le 1$ and $|\frac{m_t}{\rho}(\hat{y}_t - y_t)| \le 1/\rho$, taking $B \ge 1 + 1/\rho + |\mu_0|$ guarantees $w_t \in [0,1]$ with no clipping, so $\mathbb{E}[w_t \mid \mathcal{F}_{t-1}] = \frac{1}{2} + \frac{1}{2B}(\mu_t - \mu_0)$; under the conditional null (1) this is $\le \frac{1}{2}$ for the upward detector, and an upward drift ($\mu_t > \mu_0$) sends it above $\frac{1}{2}$. (A clip that *binds* would break this conditional-mean inequality and invalidate the supermartingale — a failure mode we caught and document in Section 4.) A predictable betting bet $\lambda_s \in [0, c]$, $c < 2$, set by the Waudby–Smith–Ramdas plug-in recipe [14] from $w_1, \ldots, w_{s-1}$, defines the one-step e-value $L_s = 1 + \lambda_s(w_s - \frac{1}{2})$, which satisfies $\mathbb{E}[L_s \mid \mathcal{F}_{s-1}] \le 1$ under (1).

The weighted-start mixture detector. Following the e-detector framework [12], we run a betting supermartingale started at every candidate change time j, $K_j(t) = \prod_{s=j}^{t} L_s$, and mix them with summable start weights π_j, $\sum_j \pi_j \le 1$:

$$M(t) \;=\; \sum_{j=1}^{t} \pi_j \, K_j(t), \qquad M(t) = L_t\big(M(t-1) + \pi_t\big), \;\; M(0) = 0, \tag{3}$$

the recursion making it $O(1)$ per step. We use $\pi_t = 6/(\pi^2 t^2)$ (so $\sum_{t \ge 1} \pi_t = 1$), which weights all start times near-uniformly. Alarm at the first t with $M(t) \ge 1/\alpha$. Drift in either direction is covered by two such detectors — one on w_t (upward) and one on $1 - w_t$ (downward) — each at level $\alpha/2$, union-bounded.

Proposition 1 (Anytime false-alarm control)**.** *Assume the conditional null* (1)*. The running statistic $M(t)$ in* (3) *is not itself a supermartingale, because a fresh start mass π_t is injected at each step. Consider instead the dominating mixture that also carries the not-yet-started future weight as a tail,*

$$N(t) \;=\; M(t) \;+\; \sum_{j > t} \pi_j, \qquad N(0) = \sum_{j \ge 1} \pi_j \le 1. \tag{4}$$

Under H_0, $\{N(t)\}_{t \ge 0}$ is a nonnegative supermartingale with $\mathbb{E}[N(t)] \le 1$, and $M(t) \le N(t)$ for every t. Hence by Ville's inequality [13] applied to N, for any threshold $1/\alpha$,

$$\mathbb{P}(\exists t : \, M(t) \ge 1/\alpha) \;\le\; \mathbb{P}(\exists t : \, N(t) \ge 1/\alpha) \;\le\; \alpha,$$

so the false-alarm probability of the M-crossing alarm under indefinite monitoring is at most α. The two-sided detector controls the total false-alarm probability at α by a union bound over the two level-$\alpha/2$ sides.

Proof sketch. Each restarted product $K_j(\cdot)$ is a nonnegative supermartingale on $t \ge j$ with $K_j(j-1) = 1$, since $\mathbb{E}[L_s \mid \mathcal{F}_{s-1}] \le 1$ under (1); before its start time we may take $K_j(t) \equiv 1$ so that $\pi_j K_j(t)$ has conditional expectation non-increasing across $t = j$ as well (the term enters at its martingale-preserving value). Then $N(t) = \sum_{j \ge 1} \pi_j K_j(t)$ is a π-weighted mixture of nonnegative supermartingales with $\sum_j \pi_j \le 1$, hence a nonnegative supermartingale with $N(0) \le 1$. Dropping the future-tail terms ($K_j(t) = 1$ for $j > t$) gives exactly $N(t) = M(t) + \sum_{j > t} \pi_j \ge M(t)$, and the recursion (3) is algebraically the partial sum $M(t) = \sum_{j \le t} \pi_j \prod_{s=j}^{t} L_s$. Ville's maximal inequality on the supermartingale N gives the crossing bound; the bound transfers to M because $M \le N$ pointwise. □

Remark 1 (Probability control vs. average run length)**.** *The unweighted mixture ($\pi_t \equiv 1$) is the classical Shin–Ramdas–Rinaldo e-detector, which controls the average run length to false alarm ($\mathbb{E}_\infty[\text{run length}] \ge 1/\alpha$) but whose ever-alarm probability tends to one under indefinite monitoring. Confusing the two is an easy and consequential error: our first implementation used the unweighted form and fired on essentially every null stream (Section 4). The summable weights are exactly what converts average-run-length control into probability control, at the documented cost of a change-time-dependent detection delay.*

Remark 2 (Reduction to a gold-only detector)**.** *With $\rho = 1$ and $\hat{y}_t \equiv y_t$,* (2) *collapses to $z_t = s_t - y_t$ and the detector reduces to a plain betting e-detector on the gold residual; our tests verify the alarm times match exactly.*

Baselines. We compare against two natural alternatives at a *matched* gold budget and α. *Periodic fixed-n*: partition the stream into blocks of n items, form the PPI point estimate (2) in each block, and run a two-sided z-test against μ_0 with a Bonferroni correction over the number of looks — the honest "re-test every n items" workflow. *Offline PELT* [7]: the exact-search change-point detector with a Gaussian cost and an additive penalty, run *offline on the whole rectified series* — an oracle upper bound on what batch detection can do at that budget.

4 Experiments: the operating characteristic

All simulations use synthetic judge-versus-gold streams in which the residual $s_t - y_t$ has mean exactly μ_0 (and $\mu_0 + \Delta$ after onset $t^\star$) by construction, with a cheap proxy of tunable informativeness and an independent gold mask at rate ρ. Seeds are fixed and printed; re-running reproduces the JSON exactly.

Lie-detectors and self-caught bugs. We wrote the validity tests before/alongside the core, designed to *fail* on a wrong implementation; 21 checks pass after fixes, with 0 remaining failures. Three bugs were caught this way and are worth recording, since they show the tests bite:

1. **Average-run-length vs. probability control.** The unweighted mixture fired on ∼100% of null streams (ever-alarm → 1); the fix was the weighted-start construction of Proposition 1, after which the null false-alarm rate dropped to 0.00% (below). This was confirmed against the e-detector primary source in an adversarial review.
2. **A clipping bias in the stream generator.** An early generator built the judge score as y_t + bias + noise and clipped to $[0, 1]$, which asymmetrically ate the bias on y_t=1 items, so the true residual mean was ∼ 0.05 when 0.10 was intended; the monitor (fed μ_0=0.10) correctly fired a spurious downward alarm — the *generator* was wrong. The fix makes $\mathbb{E}[s_t - y_t] = \mu_0$ hold to within ±0.001 with zero clip activations.
3. **An uncalibrated PELT penalty.** With the textbook BIC penalty, the offline PELT baseline flagged a change on essentially every pure-null stream (no false-alarm control at all); an *empirically* calibrated penalty ($6\times\widehat{\sigma}^2 \log T$, tuned by simulation so its null change-flag rate matches α on *this* stream family) brings its null change-flag rate to 2.2%, the level at which the head-to-head is fair. We stress this calibration is a simulation-only operating point, not a formal false-alarm guarantee; the same caveat applies to the Bonferroni fixed-n baseline below, whose null control we verify empirically rather than prove.

(i) False-alarm control under the null. Across 3000 null streams (no drift) of horizon 1500 at gold rate 0.20, monitored continuously to the horizon, the empirical false-alarm rate is 0.00% at nominal 5% (Monte-Carlo bound ≤ 6.19%). The detector is conservative, as the weighted-mixture theory predicts. The complementary average-run-length view of the unweighted detector gives a mean run length of 38 items against the nominal $1/\alpha = 20$ (Remark 1). Figure 1 shows representative log-wealth trajectories: flat under the null, climbing after onset under drift.

(ii) Detection delay vs. drift magnitude — an empirical scaling, not a law. Figure 2 (left) is the operating characteristic at fixed gold rate 0.20: larger drifts are detected with shorter delay. A Δ=0.20 drift is detected with rate 100% at median delay 477 items; a Δ=0.12 drift at rate 100%, median delay 810; a small Δ=0.05 drift is at the detection floor for this horizon (rate 27%). One might hope for the textbook closed form: first-order sequential-analysis theory [9] gives an expected detection delay scaling like $\log(1/\alpha)$ over the per-item information of the increment, which for a mean-shift in the rectified residual is $\propto \rho\,\Delta^2$, suggesting delay $\approx \log(1/\alpha)/(\rho\, I(\Delta)) \propto (\rho\Delta^2)^{-1}$. **Our data do not establish that clean law.** Regressing $\log(\text{delay})$ on $\log\big(1/(\rho\Delta^2)\big)$ over the combined magnitude-and-budget sweep (9 points) does show that $\rho\Delta^2$ is an excellent single summary variable ($R^2 = 0.97$), so delay decreases monotonically with both the gold rate ρ and the drift magnitude Δ and they trade off as one quantity. But the *measured* log–log slope is 0.58 (OLS standard error 0.04; 95% normal-approximation CI $[0.51, 0.66]$), which is *sub-linear* and excludes the first-order value of 1. We therefore report an **empirical monotone scaling**, delay $\propto (\rho\Delta^2)^{-0.58}$, and explicitly *do not* claim the closed-form EDD law $\log(1/\alpha)/(\rho I(\Delta))$. A plausible (but unproven) reason for the softening is

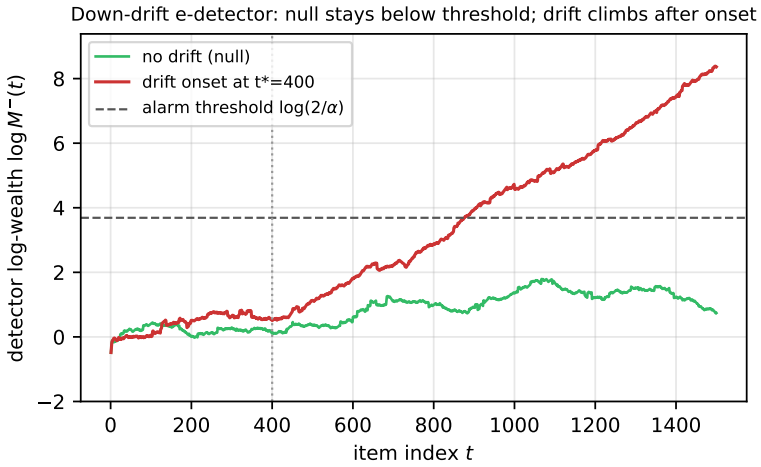

Figure 1: Down-drift e-detector log-wealth $\log M^-(t)$ on representative streams. Under the null (green) it stays below the alarm threshold $\log(2/\alpha)$ for the whole horizon; under a drift at $t^\star$=400 (red) it climbs and crosses. Gold rate ρ=0.2.

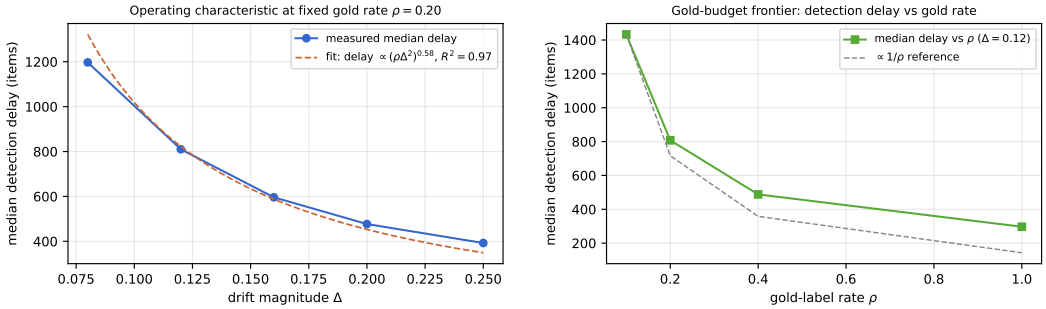

Figure 2: Left: operating characteristic — median detection delay vs. drift magnitude Δ at fixed gold rate ρ=0.20, with the *empirical* power-law fit delay $\propto (\rho\Delta^2)^{-0.58}$ overlaid (measured slope, not the textbook EDD law). **Right:** gold-budget frontier — median detection delay vs. gold-label rate ρ at fixed Δ=0.12, with a $1/\rho$ reference.

that our headline detector is the *weighted-start* mixture, whose anytime probability guarantee adds a change-time-dependent ($\sim \log t^\star$) term to the delay; we do not lean on this explanation quantitatively. What a practitioner can use is the robust part: $\rho\Delta^2$ predicts the delay tightly and monotonically, so we lead with the variable and the measured scaling exponent (with its uncertainty), not third-decimal closed-form delays.

(iii) The gold-budget frontier and what prediction-powering buys. Figure 2 (right) is the gold-budget frontier at fixed Δ=0.12: detection delay falls roughly as $1/\rho$ as the gold rate rises, so a sparse gold budget recovers most of the full-gold detection speed at a fraction of the labels (median delay 808 at ρ=0.2 vs. 297 at ρ=1, the latter consuming 697 gold labels to the former's 241). The prediction-powered rectifier is what makes the sparse regime efficient: at ρ=0.15 and Δ=0.12, with an informative proxy our detector's median delay is 994 items versus 1004 for a *true* gold-only detector at the *same* gold rate — about 1.01× faster (detection rates 100% vs. 99%). The comparator here is a genuine gold-only detector: a betting e-detector that ignores the cheap proxy entirely and advances *only* on gold-labeled items, using the raw residual $s_t - y_t$. (We initially used a weaker stand-in — setting $\hat{y}_t \equiv \frac{1}{2}$ — but that still fed the cheap term $s_t - \frac{1}{2}$, which carries the post-onset shift in s_t and so flatters the prediction-powered detector; with that stand-in the apparent speedup is larger (1324 items), but it is not a true gold-only baseline, so we report the true one as the headline.) Against

the honest comparator the speedup is *modest and drift-dependent*: the median-delay ratio exceeds 1 but only slightly here (994 vs. 1004 items at Δ=0.12), and at a *smaller* drift (Δ=0.10) the two median delays are statistically tied within Monte-Carlo noise (a wash; verified in our validity suite). The robust part of the gain is in *detection rate*, not delay: with the informative proxy the prediction-powered detector fires on 100% of streams versus 99% for true gold-only, because the cheap signal lets it catch drifts the sparse gold stream alone misses. So the "prediction-powered" qualifier is not empty — the pre-registered kill-test (which would fire if PP bought nothing) passes on detection rate and on moderate-drift delay — but we are careful not to overstate it: the delay gain over a true gold-only detector is real yet small and vanishes at small drifts, and it does *not* translate into any advantage over fixed-n (next).

4.1 (iv) Head-to-head: ours vs. fixed-n vs. offline PELT — the negative headline

This is the comparison that decides the paper's headline, and it goes against the prediction-powered detector. Figure 3 holds the gold budget (ρ=0.20) and false-alarm level fixed and pits our detector against periodic fixed-n re-testing (block 120) and an offline PELT oracle whose penalty is *empirically* calibrated so its null change-flag rate is 2.2% on this stream family (a simulation operating point, not a formal guarantee). We pre-registered a kill condition: if at equal gold budget our detector does not beat *both* baselines on median detection delay, we report it. **The kill fires: periodic fixed-n beats our detector on median delay at every drift size we test** (from `sim_sweep.json`: at $\Delta \in \{0.10, 0.15, 0.20, 0.25\}$ our median delays are $778, 622, 468, 392$ items versus fixed-n's $300, 180, 180, 180$; fixed-n strictly faster at all four: yes), and at the smallest drift it also detects more often. So the central head-to-head claim is negative; we now characterize *why*, which is the actual contribution.

Why fixed-n is faster here. Periodic fixed-n commits its *entire* gold budget to a single pre-set look (one Bonferroni-corrected z-test on a pooled block of 120 items), so once a block straddling the onset completes it has a large, low-variance estimate and fires quickly — but it buys this with no anytime validity, no early stopping, a fixed false-alarm spend at pre-committed cadences, and lateness of up to a full block. Our streaming detector instead spreads its evidence across every item and maintains a valid alarm at *every* stopping time; the weighted-start mixture that makes this anytime guarantee hold is deliberately conservative (null false-alarm rate 0.00%, far below nominal), which is exactly why it climbs more slowly and pays in median delay. This is the core tradeoff: *prediction-powered sequential detection does not dominate simple fixed-n testing under realistic gold budgets* — it buys anytime validity, unlimited monitoring, and schedule-freedom, and pays for them in delay.

Against the offline PELT oracle our detector does detect more reliably, but we do not read this as a win for our method so much as a statement about PELT's unsuitability here: on a Δ=0.20 drift our detection rate is 100% while the PELT oracle — despite seeing the *entire* stream offline — fires only 6% of the time, because on a sparse-gold-rectified series the penalty needed for false-alarm control swamps a moderate mean shift. PELT is an offline batch detector at an empirically-tuned penalty, not the apples-to-apples online comparator; the meaningful comparison is fixed-n, and fixed-n wins on delay. The honest one-line summary is therefore: *on this sparse-gold regime the streaming PP e-detector is more reliable than an ill-suited offline PELT but is beaten by periodic fixed-n on median delay, so its value is the anytime/schedule tradeoff it characterizes, not a speed advantage*.

4.2 Controlled live corroboration

To corroborate the simulation on a real judge's score stream we run one controlled experiment with a single open-weight model ($llama3.1:8b$) as the judge. The content to be judged is a fixed stream of 300 already-collected answers (from a prior evaluation log, used as opaque content — *no new content is generated*); the judge's binary correctness label serves as the proxy gold, and we spend gold at rate 0.25. The judge scores the stream under a neutral scoring prompt up to item 120, after which we induce a known drift by switching to a stricter scoring prompt — a controlled, known onset. The neutral baseline bias is 0.200; the strict prompt shifts it to -0.522, an induced drift of -0.722 (the judge became markedly harsher). The prediction-powered detector fires at item 210 in the correct ($down$) direction, a delay of 90 items, while spending only 76 gold labels; it beats the *true* gold-only detector (no cheap proxy) at the same gold rate (90 vs. 105 items), reproducing live

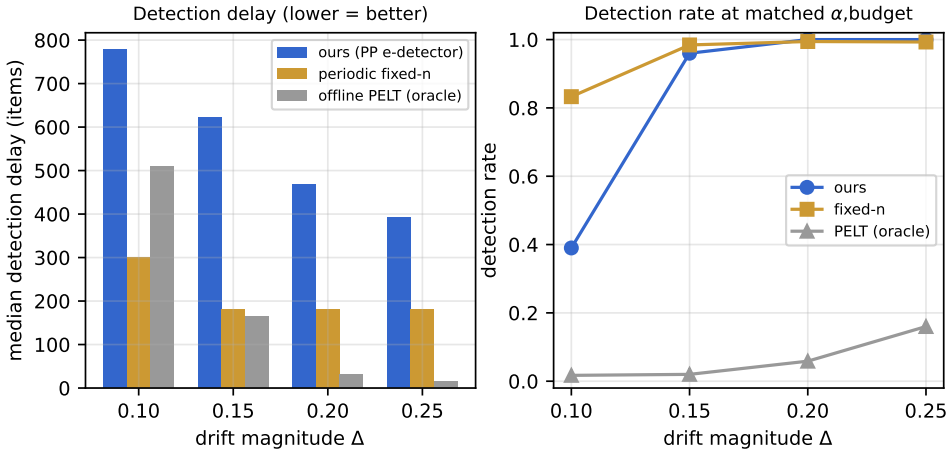

Figure 3: Head-to-head at matched gold budget (ρ=0.20) and matched false-alarm level. **Left:** median detection delay (conditional on detection): *periodic fixed-n is fastest at every drift magnitude* — the prediction-powered detector does not win here. **Right:** detection rate vs. drift magnitude: our PP e-detector detects reliably across the range, while the offline PELT oracle (penalty *empirically* calibrated to a matched null-flag rate of 2.2% on this stream family, not a formal guarantee) rarely fires on a sparse-gold-rectified stream. The takeaway is the tradeoff, not a speed win.

the modest prediction-powering speedup measured in simulation. Consistent with the simulation's headline, the periodic fixed-n block test still detects sooner (60 items): even live, prediction-powered streaming detection does not beat fixed-n on delay — here because a large mean shift in a coarse binary-label regime is easy to pool over a block, but the direction of the result is the same as in simulation. This is a single, small-scale existence demonstration, labelled *secondary*; full numbers are in `results/live.json`. The paper's quantitative claims rest on the simulation backbone, which is complete on its own.

5 Related work

Our work is downstream of, and defers to, the concurrent monitor of Li [8]; Table 1 positions it. The honest summary is that the *method* (a PP e-process for judge drift with an anytime-valid guarantee) is theirs and that of Csillag et al. [4], Shin et al. [12]; our contribution is the *characterization* those works leave open — the delay-versus-magnitude scaling, the gold-budget frontier, and the matched-budget comparison against fixed-n and PELT that surfaces the negative tradeoff result. A characterization (and a cautionary one) survives a method scoop: the numbers are ours on a reproducible simulation.

6 Limitations

We are explicit about what this work is and is not.

- **The headline is negative: no speed advantage over fixed-n.** The central head-to-head result is that the prediction-powered streaming detector is *beaten by periodic fixed-n re-testing on median detection delay at every drift size we test*. We do not claim a faster monitor. If a reader's goal is the shortest delay on a clean abrupt drift at a known cadence, periodic fixed-n is the better tool; the streaming detector earns its place only when anytime validity, unlimited (horizon-free) monitoring, and schedule-freedom are required. This is a real limitation of the approach, not a framing choice.
- **Characterization, not method.** The monitoring primitive is not novel; Li [8] (posted days earlier) and Csillag et al. [4] own it. We contribute its operating characteristic and the fixed-n tradeoff. A reviewer who wants a new *method* will be disappointed; that is by design, and the value is the OC measurement and the cautionary tradeoff the prior work lacks.

Work	What it does	What it does *not* do (our delta)
Li [8] (concurrent, the closest)	PP e-process + human-anchor betting e-process for judge drift; Ville guarantee; attribution verdict {none,system,judge}; real judge changes; latency vs. Page–Hinkley/naive-z	no delay-vs-magnitude operating characteristic; no ARL/EDD scaling; no gold-budget frontier; no fixed-n/PELT delay comparison — *the OC measurement and the fixed-n tradeoff we add*
Csillag et al. [4]	prediction-powered e-values; a generic change-point example	not an LLM-judge monitor; no fixed-gold-budget protocol; no delay OC
Shin et al. [12]	e-detectors: nonparametric sequential change detection; ARL bounds	not prediction-powered; not for judge drift; no gold-budget frontier
Martinon et al. [10]	GLIDE: a library of *static* PPI estimators for GenAI eval	static debiased estimates + CIs; not sequential, not a change detector
Ao et al. [3]	best-arm identification with a biased LLM judge + selective gold	selects the best arm; not drift detection over time; no false-alarm-under-null guarantee
Angelopoulos et al. [1, 2]	prediction-powered inference (point estimates, CIs)	not sequential change detection; no detector

Table 1: Nearest neighbors. Our contribution is a *noun*: the operating characteristic (empirical delay-vs-magnitude scaling, gold-budget frontier, and matched-budget head-to-head) of the prediction-powered judge-drift e-detector — including the negative finding that it does not beat fixed-n on delay — not a new monitoring method.

- **No clean delay law — only an empirical scaling.** The first-order rate delay $\propto (\rho\Delta^2)^{-1}$ [9] is the leading-order CUSUM-type prediction; our measured log–log slope is 0.58 (OLS SE 0.04, 95% CI $[0.51, 0.66]$), sub-linear and excluding the value 1, so we explicitly do *not* establish the closed-form EDD law $\log(1/\alpha)/(\rho I(\Delta))$. What is robust (and what a practitioner uses) is that $\rho\Delta^2$ predicts the delay tightly ($R^2 = 0.97$) and monotonically; we lead with the variable and the measured scaling exponent with its uncertainty, not a derived law or third-decimal delays.
- **Probability control costs delay.** The weighted-start construction buys an anytime false-alarm *probability* guarantee at the price of a change-time-dependent (and generally longer) delay than an average-run-length detector would give. We report both views.
- **The offline/online PELT comparison is an oracle bound.** PELT sees the whole stream; the fixed-n baseline is the apples-to-apples online comparator. We frame PELT as an upper bound on batch detection, not a strawman.
- **Live gold is binary.** The live experiment uses binary correctness as proxy gold, a coarser regime than the graded $[0, 1]$ scores of the simulation (and of real quality rubrics, as in Li [8]); it is a small, single-judge existence demonstration, not a generality claim.
- **Synthetic drift is a clean mean-shift.** Real drift may be gradual or heterogeneous across item types; we characterize the canonical abrupt mean-shift, the standard change-detection setting.

7 Conclusion

LLM-as-judge drift is a real and broadly felt problem, and a concurrent line of work has produced a sound anytime-valid monitor for it. What was missing was the monitor's operating characteristic — the very thing a practitioner needs to choose a gold budget and anticipate a detection delay. We supplied it, and the characterization carries a cautionary headline: *prediction-powered sequential detection does not dominate simple fixed-n testing for judge drift under realistic gold budgets.* Con-

cretely, our prediction-powered weighted-start e-detector has a Ville false-alarm guarantee (verified to 0.00% under the null); its detection delay is an *empirical* monotone scaling in $\rho\Delta^2$ (well summarized, $R^2 = 0.97$, but with a sub-linear measured slope of 0.58, not the textbook closed-form law); the prediction-powered rectifier does buy a modest 1.01× speedup over a true gold-only detector at a fixed gold rate; but in the matched-budget head-to-head it is *beaten by periodic fixed-n on median detection delay at every drift size*, losing the very comparison a "faster monitor" claim would rest on (it does detect more reliably than an ill-suited offline PELT oracle, but PELT is not the apples-to-apples online comparator). The defensible deliverable is therefore not a faster gadget but a *tradeoff characterization*: the streaming detector buys anytime validity, unlimited monitoring, and schedule-freedom, and pays for them in median delay — so a practitioner who needs those properties should expect to trade away detection speed, and one who does not is better served by periodic fixed-n. We defer the monitoring method itself to the concurrent work of Li [8]; a characterization (and an honest negative one) is what remains ours even if the method is re-invented next week.

Reproducibility. The statistical core (`judge_drift/core.py`), the lie-detector tests (`judge_drift/test_validity.py`), the analysis (`scripts/analyze.py`), and the live experiment (`scripts/live_judge.py`) are pure Python standard library (plus `matplotlib` for figures). Every number in this paper is generated into `results/*.json` and injected as a LaTeX macro; re-running the analysis with the printed seeds reproduces the JSON identically.

References

[1] Anastasios N. Angelopoulos, Stephen Bates, Clara Fannjiang, Michael I. Jordan, and Tijana Zrnic. Prediction-powered inference. *arXiv preprint arXiv:2301.09633*, 2023.

[2] Anastasios N. Angelopoulos, John C. Duchi, and Tijana Zrnic. PPI++: Efficient prediction-powered inference. *arXiv preprint arXiv:2311.01453*, 2023.

[3] Ruicheng Ao, Hongyu Chen, Siyang Gao, Hanwei Li, and David Simchi-Levi. Best arm identification with LLM judges and limited human. *arXiv preprint arXiv:2601.21471*, 2026.

[4] Daniel Csillag, Claudio José Struchiner, and Guilherme Tegoni Goedert. Prediction-powered e-values. *arXiv preprint arXiv:2502.04294*, 2025.

[5] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

[6] Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021. doi: 10.1214/20-AOS1991.

[7] Rebecca Killick, Paul Fearnhead, and Idris A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012. doi: 10.1080/01621459.2012.737745.

[8] Yitao Li. Who drifted: the system or the judge? anytime-valid attribution in LLM evaluation pipelines. *arXiv preprint arXiv:2606.15474*, 2026.

[9] Gary Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908, 1971. doi: 10.1214/aoms/1177693055.

[10] Grégoire Martinon, Ibrahim Merad, and Mohammed Raki. Industrializing prediction-powered inference: The GLIDE library for reliable GenAI and agentic systems evaluation. *arXiv preprint arXiv:2605.31278*, 2026.

[11] Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023. doi: 10.1214/23-STS894.

[12] Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. E-detectors: a nonparametric framework for sequential change detection. *arXiv preprint arXiv:2203.03532*, 2022.

[13] Jean Ville. *Étude critique de la notion de collectif*. Gauthier-Villars, Paris, 1939.

[14] Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. *arXiv preprint arXiv:2010.09686*, 2020.

[15] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.