ANOTHER SAMPLE OR A BETTER VERIFIER? WHAT INFERENCE BUDGET BUYS FOR CONSISTENT AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent that handles refunds is useful only if it gets them right every time, not most of the time. Benchmarks call this consistency and measure it as pass^k, the chance that all k attempts at a task succeed. We ask what a fixed byte budget per conversation should buy to raise it, a sample, a verifier call, a vote, or a retry, and we find a rule. At the budgets operators run, how cheaply a candidate is selected matters ten times more than how accurately; we call this the cost-over-accuracy rule. The evidence comes from replaying 6,428 released τ-bench and τ^2-bench trajectories through an evaluator that scores any allocation policy exactly, running no model. Across four frontier models and three domains, a single attempt succeeds 56.3% of the time and all four attempts 41.4%. Buying operations rather than samples raises pass^4 by +1.76 points at equal cost. A perfect verifier that must read every candidate reaches 43.6%, no better than a vote, because reading costs as much as sampling and cuts the draws in half; the same verifier made free reaches 53.2%, and a proposition predicts where the two cross. A verifier that reads only a candidate's write actions, 2% of the bytes, follows the rule and reaches 47.7%. Two expectations written down before the study were refuted. Optimizing for consistency changed no allocation, and a learned controller matched but did not beat a tuned rule. Everything replicates on a second release.

1 INTRODUCTION

Agents built on language models now complete refunds, rebook flights, and change telecom plans on a customer's behalf. A mistake there is an action, not a sentence, so it is costly to undo, and the operator has one lever that does not require a new model, namely how much inference to spend on each conversation and on what. The literature on test-time compute has shown that spending more can substitute for a larger model (Snell et al., 2025; Wu et al., 2025; Brown et al., 2024), and it has settled on two ways to spend, more samples or a better verifier, studied on single-turn problems with accuracy as the goal. This paper measures what those purchases are worth for a deployed agent, and the answer inverts the usual priority. A deployed agent is judged by whether it does the task every time it is asked, which is what we mean by a consistent agent, not by its average accuracy. At the budgets operators run, what the selection step costs matters far more than how accurate it is; a perfect selector is worth about one point over a cheap one when it must read the candidates and about ten when it reads them free. We call this pattern the cost-over-accuracy rule and test it in Section 5.

To measure consistency we use pass^k, the metric τ-bench (Yao et al., 2025) introduced, the probability that all k independent attempts at a task succeed; its authors introduced it because agents competent on one attempt proved inconsistent across attempts. The gap is large. Across the four frontier models Sierra released with τ^2-bench (Barres et al., 2025), first-attempt success averages 56.3% and success on all four attempts 41.4%; every cell, one per model and domain, loses a fifth or more by $k{=}8$ and the weakest model over two fifths by $k{=}4$ (Figure 1b). The decay is expected, p^k under independent trials; what the released agents show is its size. The loss is concentrated. Between 31% and 51% of tasks in every model and domain succeed on some trials and fail on others (Figure 1b); tasks that always pass keep their value at every k, tasks that never pass have none, and it is the sometimes-tasks whose contribution decays as p^k. If every sometimes-task were pushed to certainty, pass^4 would rise to 74.8% on average, which is the ceiling on what inference-time allocation can buy at a fixed model.

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.

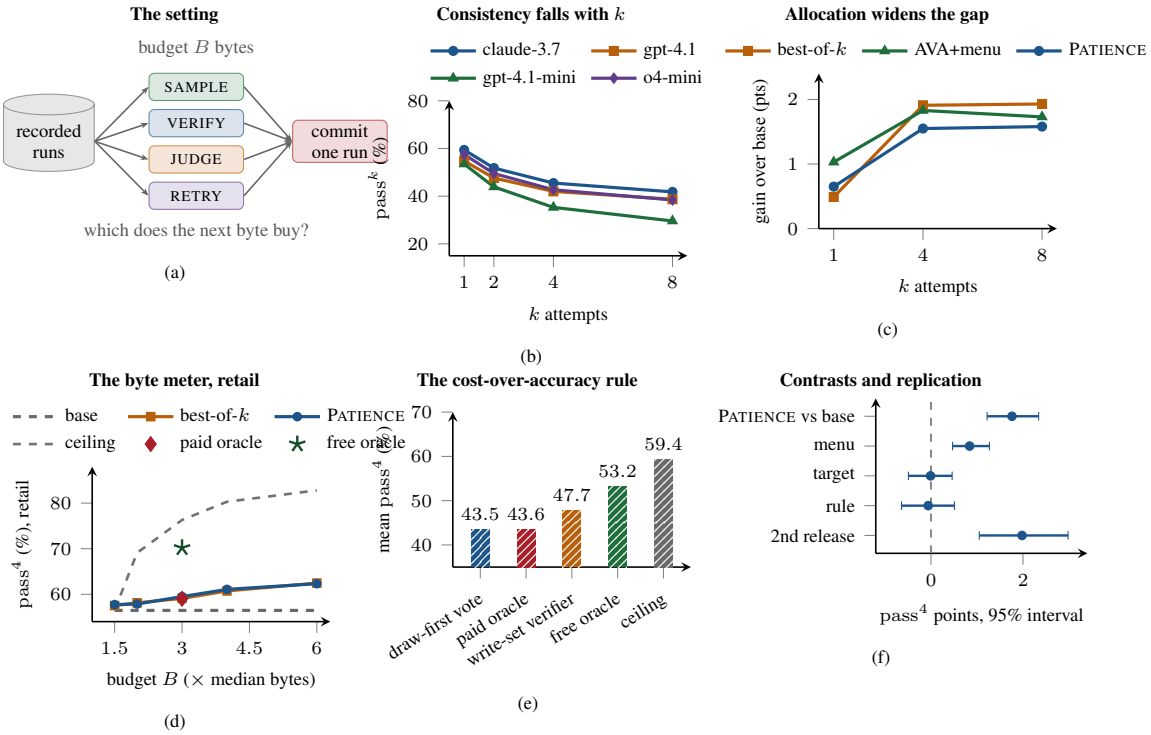

Figure 1: The paper in six panels, (a) the setting, (b) pass^k of the released agents against k, (c) gain over the base agent against k over the twelve cells, one per model and domain, (d) pass^4 against budget on retail with the two perfect-selection points and the ceiling of Theorem 1, (e) the cost-over-accuracy rule, with the draw-first vote as the cheap selector and the write-set verifier as the one it recommends, and (f) the paired contrasts with 95% intervals.

So what should the operator spend on? Figure 1a shows the choice. A run can buy four things. A SAMPLE adds a candidate trajectory; a VERIFY scores candidates and prunes; a JUDGE votes among candidates and commits the plurality; a RETRY re-executes a step that failed with a visible tool error. They differ in byte cost, bytes being what serving and billing layers record, and in effect, since each one's value depends on what has already been bought. The closest prior controller, Anytime Verified Agents (AVA) (Patel, 2026), allocates across sampling and verification under a token budget and aims at accuracy; it has no vote and no retry. Recent ICLR work sharpens pieces of this picture (Huang et al., 2026; Dorner et al., 2026; Komiyama et al., 2026; Ma et al., 2026; Dong et al., 2026), each fixing the kind of spending in advance.

Two obstacles have prevented a clear answer to this question, cost and confounding. Live experiments at scale are expensive and depend on provider state on the day they run, and a verifier or judge the authors introduce can quietly do the allocator's work. Our route removes both. Sierra released the trajectories behind its τ^2-bench results, four per task, with tool calls, error flags, and graded outcomes. We treat a task's four trajectories as the pool a run draws from, define each operation on that pool with a byte cost read from the transcripts, and compute the success probability of any allocation policy exactly, by recursion over the multiset of draws (Section 3). The verifier is a classifier on observable features trained out of fold; the judge is a plurality vote over canonical write-action sets, the state-changing tool calls a run makes, which is self-consistency (Wang et al., 2023) applied to actions; the retry fires only on an observable tool error. No decision sees a label.

We then run the full experiment, and Figures 1c to 1f preview its findings. The gap over the base agent grows from $k{=}1$ to $k{=}4$, the budget buys consistency only for rules that vote, and the paid perfect verifier lands on the voting curves while free selection lands far above them. The experiment covers twelve model-domain cells from τ^2-bench, a five-point budget sweep, and three verifiers up to an oracle. It compares eleven allocation rules, including a faithful AVA reimplementation, AVA given our menu, the set of four operations a run may buy, and a controller learned from exact dynamic-programming targets on held-out tasks. Every comparison is paired over 1,112 task-model pairs with

cluster-bootstrap intervals, and the study is replicated on the 1,980 original τ-bench trajectories of gpt-4o and claude-3.5-sonnet. This paper makes four contributions.

• **The cost-over-accuracy rule, and the verifier it recommends.** A draw-first rule with the label itself as verifier reaches 43.6% mean pass^4, no better than a vote, because verification costs what a candidate costs and cuts the draws in half; the same selection made free reaches 53.2%. A verifier that reads only the write-action set, 2% of the bytes, reaches 47.7% (Section 5). For every rule that votes, one more draw beats a flawless verifier (Table 2).

• **Operations beat samples, and the menu is the lever.** Every allocator beats the base agent at equal bytes. Best-of-k gains +1.94 points of pass^4 [+1.39, +2.48] and PATIENCE, the learned controller, +1.76 [+1.22, +2.34], with the gap growing from k=1 to k=4 as Proposition 1 predicts (Figure 1c). AVA given JUDGE and RETRY beats AVA by +0.84 points [+0.47, +1.27]. PATIENCE confirms the rule rather than beating it; what it learns is to sample early and vote late.

• **Two negative results.** Tuning for pass^k instead of accuracy selected identical thresholds in all twelve cells and left PATIENCE unchanged (-0.01 [-0.49, +0.46]). Learning matched but did not beat a tuned rule (-0.06) or voting (-0.18), though it transfers across domains at parity.

• **An exact, public evaluator, replicated.** Any allocation policy over recorded agent trajectories is scored exactly in minutes on one core with no model queried, and the findings hold on a second release with different models and a different user simulator.

We do not claim a new state of the art. The gains are about two points of pass^4, roughly 4% relative, and Section 5 says what stands between two points and ten.

2 PROBLEM AND A PREDICTION

This section fixes the objects the study measures and states the prediction Section 5 tests.

Runs, pools, budgets. A task τ is a customer request in a domain with a tool API and a written policy. A run is a conversation with a simulated user that ends in a graded outcome and consumes bytes of prompt and completion. The operator fixes a byte budget B per run. In replay a task comes with a pool of recorded runs, and a policy run draws from it.

Operations. The menu is $\mathcal{O} = \{\text{SAMPLE}, \text{VERIFY}, \text{JUDGE}, \text{RETRY}\}$. SAMPLE draws one more trajectory into the candidate set. VERIFY scores every candidate with a classifier and marks the set as scored. JUDGE votes over the candidates' write-action sets and commits the plurality. RETRY is allowed once per run and only when the candidate that would be committed carries a visible tool error; it replaces that candidate with a fresh draw.

Objective and estimator. We adopt the benchmark's assumption that a task's attempts are conditionally independent given the task; under it a task with per-trial success probability p_τ has $\mathrm{pass}^k(\tau) = p_\tau^k$, and the population metric is $\mathrm{pass}^k = \mathbb{E}_\tau[p_\tau^k]$ (Yao et al., 2025). With n recorded trials and c successes the benchmark estimates $\mathrm{pass}^k(\tau)$ by $\binom{c}{k}/\binom{n}{k}$, the unbiased estimator of p_τ^k from n trials in the sense of Chen et al. (2021); the plug-in $(c/n)^k$ is biased upward, and that bias is what the split-half check in Section 5 measures.

Proposition 1 (Accuracy-maximizing and consistency-maximizing spend can differ)**.** *There exist a task distribution and two equal-cost operations o_1, o_2 such that o_1 maximizes $\mathbb{E}[\mathrm{pass}^1]$ while o_2 maximizes $\mathbb{E}[\mathrm{pass}^k]$ for every $k \geq 2$.*

Proof. Let 60% of tasks have $p = 0.9$ and 40% have $p = 0.2$. Let o_1 lift the hard tasks to $p = 0.5$ and o_2 lift the easy tasks to $p = 1$. Then $\mathbb{E}[\mathrm{pass}^1 \mid o_1] = 0.74 > 0.68 = \mathbb{E}[\mathrm{pass}^1 \mid o_2]$, while $\mathbb{E}[\mathrm{pass}^2 \mid o_1] = 0.586 < 0.616 = \mathbb{E}[\mathrm{pass}^2 \mid o_2]$, and the second ordering holds for all $k \geq 2$ because $p^k \to 0$ for every $p < 1$ and stays at 1 for $p = 1$. □

In words, an operation that lifts hard tasks a little can win on accuracy while one that makes easy tasks certain wins on consistency, because a task at $p = 1$ never decays with k.

Theorem 1 (Ceiling under a draw budget)**.** *If a run draws at most m candidates from a pool in which each draw succeeds independently with probability p_τ, then every commit rule, including one that sees the labels, succeeds with probability at most $1 - (1 - p_\tau)^m$, so $\text{pass}^k \le \mathbb{E}_\tau\big[(1 - (1 - p_\tau)^m)^k\big]$. Equality holds exactly when the rule draws m times and commits a passing candidate whenever one was drawn, which no rule that pays per candidate for its selection can do under a budget that admits m draws.*

Proof. A committed candidate passes only if some drawn candidate passes, and the draws are independent, so the failure probability is at least $(1 - p_\tau)^m$, with equality when all m draws are made and selection is perfect. Paying per candidate for selection spends budget that would have bought draws, so fewer than m are made. □

In words, no selector, however good, can rescue a run that never drew a passing candidate. The theorem fixes what a number of draws can buy and says nothing about how well a selector uses them; Section 5 measures that use as selection efficiency.

Proposition 2 (Breakeven read cost)**.** *Let a budget admit m draws when selection is free, write $C(m) = \mathbb{E}_\tau[(1-(1-p_\tau)^m)^k]$ for the ceiling of Theorem 1, and let a verifier charge a fraction ρ of a candidate's bytes to read it. Then perfect selection through that verifier reaches at most $C(m/(1+\rho))$. A cheap selector that captures a share e of the ceiling's gain over the base agent at m draws beats it whenever ρ exceeds the ρ^* that solves $C(m/(1+\rho^*)) = \text{pass}^k(\textit{base}) + e\,[C(m) - \text{pass}^k(\textit{base})]$.*

Proof. Each read costs ρ of a draw, so a budget of m draws buys $m/(1+\rho)$ draws with their reads; apply Theorem 1 at that number and compare with the cheap selector's value. □

The proposition turns the rule into a prediction. Given the pools, it fixes the curve a paid perfect verifier traces as its read cost falls, and the read cost at which a cheap selector of known efficiency ties it. Section 5 checks both.

Proposition 1 carries a prediction and a condition. The prediction is that operations which move a sometimes-task to always-pass gain value relative to sampling as k grows, so any allocator's advantage over the base agent should widen with k. The condition says when the two objectives can disagree at all. Write the effect of operation o as a lift from p_τ to $p_\tau^o \ge p_\tau$. If one operation dominates the other on every task, $p_\tau^{o_1} \ge p_\tau^{o_2}$ for all τ, then $\mathbb{E}[(p^{o_1})^k] \ge \mathbb{E}[(p^{o_2})^k]$ for every k, because $x \mapsto x^k$ is monotone on $[0, 1]$, and the objectives rank them identically; disagreement requires lifts that cross, one operation better on some tasks and worse on others, as in the proof. Dominance is sufficient for agreement, not necessary. Whether a real menu does that is an empirical question, and Section 5 answers it for ours.

3 Trace-Driven Evaluation

This section describes the data and the evaluator that scores every rule exactly, which is what makes the comparisons of Section 5 fair.

Data. Sierra shipped, with the τ^2-bench repository (Barres et al., 2025) (version 1.0.1, MIT license), the results of running claude-3.7-sonnet, gpt-4.1, gpt-4.1-mini, and o4-mini on retail (114 tasks), airline (50), and telecom (114), four trials per task, agent temperature 0, user simulator gpt-4.1. Each trajectory carries its messages, tool calls, tool-error flags, termination reason, and the grader's reward. We extract one row per trajectory, 4,448 in all, with only inference-time observable fields as features and the reward as a label; our re-scoring reproduces the benchmark paper's own pass^1 values to the decimal. As a second, independent release we use the historical trajectories shipped with the original τ-bench repository (Yao et al., 2025). They cover gpt-4o with four trials and claude-3.5-sonnet with eight trials per task on retail and airline, 1,980 trajectories over 330 task-model pairs. They come from a different benchmark version, user simulator, model generation, and year. Re-scoring gpt-4o on retail reproduces the published 60.4% pass^1.

Replay model. Figure 2 (appendix) shows the loop. The label enters only at scoring and at the training targets fitted on other folds. The replay model rests on one assumption, that a run of policy π on task τ draws trajectories independently and uniformly from the task's pool, the replay analogue of sampling a fresh conversation. Because draws are exchangeable and, by a design choice we hold to throughout, every policy reads them only through their multiset, the run is described by the multiset of draws, two flags (scored, voted), a retry counter, and the bytes spent. Every operation maps a state to a distribution over next states with at most four branches, so the success probability $p(\pi, \tau)$ of the committed candidate is computed exactly by recursion with memoization; nothing in the paper is sampled. We report $\mathrm{pass}^k(\pi) = \mathbb{E}_\tau[p(\pi, \tau)^k]$ for every policy, the base agent included with $p = c/4$. Two cautions apply. The plug-in form sits slightly above the finite-pool estimator for $k > 1$, equally for all methods, and Figure 1b anchors the base agent to the benchmark's estimator. And k runs of a policy on one task share the recorded pool, so $p(\pi, \tau)^k$ is the policy's consistency over that pool, as with the benchmark's own estimator. The recorded trials are the ones the benchmark's leaderboard computes pass^k from. What the replay measures is therefore a policy's consistency on the benchmark's own trial distribution, the quantity the leaderboard reports, not a proxy for live decoding.

Costs. The cost model is a design choice, stated here once. Bytes accumulate on a byte meter, the running total a run has spent. SAMPLE costs the drawn trajectory's transcript bytes; VERIFY the bytes of every candidate it reads plus one kilobyte of prompt; JUDGE half a kilobyte plus the write-action sets, since a plurality vote needs no model call; RETRY the replacement's bytes. The primary budget B is three times the cell's median transcript, about three draws; Table 3 in the appendix sweeps it from $1.5\times$ to $6\times$. A policy that cannot afford an operation commits.

Signals. The verifier is a logistic regression on tool-error count, turns, tool calls, write actions, transfer-to-human flag, transcript length, termination reason, and model identity, fit with five task-grouped folds so that no trajectory is scored by a model that saw its task. Its out-of-fold AUC is 0.63 on retail, 0.63 on airline, and 0.88 on telecom; the last is high because telecom failures are largely visible as tool errors. The judge canonicalizes every state-changing tool call (name plus sorted arguments, by the domain's own write-tool typing) and votes over the resulting sets, breaking ties by verifier score; neither signal uses the reward.

Scope. Bytes are transcript bytes, excluding the system prompt a provider re-sends each turn, identically for all methods; operations act on whole trajectories.

4 Allocation Rules and Experimental Design

This section names the rules the results compare and the design that makes every contrast paired.

Fixed recipes. The *base agent* draws once and commits. *Best-of-k* draws while the budget allows and votes. *Agreement stopping* draws until two candidates agree on their write set, then votes, a reservation rule in the spirit of Pandora's box, one that stops as soon as the value found clears a threshold (Weitzman, 1979). *Verify cascade* scores the pool and draws one more only when the best score is below 0.7. *Uniform menu* follows the fixed order SAMPLE, SAMPLE, VERIFY, JUDGE, RETRY where legal.

AVA, reimplemented. AVA (Patel, 2026) allocates across sampling and verification by an uncertainty estimate that fuses self-consistency and verifier scores, verifies in a cascade with early exit, and stops when confidence clears a threshold. On the replay menu {SAMPLE, VERIFY}, confidence is the larger of the best verifier score and the plurality share; while confidence is below θ and uncertainty exceeds τ, the rule verifies an unscored pool first, then draws, and commits the verified-best candidate. Thresholds are tuned per cell on a grid, in sample, which favors the baseline. *AVA+menu* is the same rule given JUDGE and RETRY.

PATIENCE, learned. The learned controller, which we call PATIENCE because what it learns is to keep drawing while the budget lasts and to decide late, is fitted Q-iteration with exact targets. On training-fold tasks we compute by dynamic programming the best value achievable from every

Table 1: pass^4 (%) at equal byte spend, one budget per cell; best allocator per row in bold.

Model	Domain	base k=1	base k=4	Best-of-k	Agreement stopping	Verify cascade	Uniform menu	AVA	AVA+menu	PATIENCE
claude-3.7	retail	78.7	65.2	66.0	65.6	65.2	**66.3**	65.3	66.2	66.1
gpt-4.1	retail	74.1	59.7	**62.2**	61.3	59.9	61.8	59.8	60.5	61.8
gpt-4.1-mini	retail	66.0	46.9	50.1	49.9	48.6	49.5	48.3	49.4	**52.0**
o4-mini	retail	71.5	54.0	58.0	57.0	55.3	57.4	55.3	56.3	**58.1**
claude-3.7	airline	50.0	38.7	39.6	39.9	39.4	39.3	39.8	**40.4**	39.8
gpt-4.1	airline	56.0	43.2	44.1	44.7	44.6	44.4	44.4	**45.2**	44.5
gpt-4.1-mini	airline	50.5	34.1	36.7	35.2	36.7	36.0	36.2	**38.2**	34.6
o4-mini	airline	59.0	44.0	46.7	46.3	44.7	**46.9**	44.2	43.6	43.5
claude-3.7	telecom	49.3	32.7	35.2	34.8	33.6	**35.2**	33.6	34.6	33.5
gpt-4.1	telecom	34.2	23.2	24.7	**24.9**	23.5	24.7	23.6	24.8	24.6
gpt-4.1-mini	telecom	43.9	24.7	25.3	25.6	26.2	25.2	26.1	**27.2**	25.1
o4-mini	telecom	42.1	30.1	30.8	30.9	32.0	31.0	32.0	**32.1**	31.5
Mean of 12 cells		56.3	41.4	**43.3**	43.0	42.5	43.1	42.4	43.2	42.9

reachable state with the task's own labels for the objective $\mathbb{E}[p^k]$, and record for each state and legal operation the gain $Q^*(s, o) - V_{\text{commit}}(s)$. A gradient-boosted regressor per operation maps thirteen observable state features (candidate counts, plurality share, distinct write sets, error on the would-be commit, scored and voted flags, verifier scores, remaining budget, predicted costs) to that gain. At test time the controller buys the legal, affordable operation with the largest predicted gain and commits when none is positive. Training pools the tasks of all four models in a domain under shared task-grouped folds, so a test task is never seen by its controller under any model. We first divided predicted gain by predicted cost, the adaptive-submodular rule (Golovin & Krause, 2011; Khuller et al., 1999); it starved SAMPLE, whose value appears only after a later JUDGE. Since the budget is inside the target, the value itself is the right decision variable.

Design and statistics. Table 7 (appendix) places the design against the studies a program committee would compare it with; the last row is the only multi-turn, byte-metered, consistency-targeted design, and the only one scored without sampling. The rules span a factorial in rule, menu, and target, so each axis has a paired contrast (Table 5, appendix). Intervals are 2,000-sample bootstraps over the 1,112 (task, model) pairs that resample the 278 task clusters, since a task appears under all four models; a star marks an interval excluding zero, and each row of Table 4 isolates one axis of the design. Half-widths near half a point at k=4 are the smallest differences this design resolves, so the null contrasts are nulls at that resolution, not proofs of equality; of the 9 contrasts reported, 6 exclude zero and all 6 survive a Holm correction. AVA's thresholds are tuned in sample, which can only help AVA; PATIENCE never sees a test task.

5 RESULTS

Five questions organize this section. Do operations beat samples at equal bytes, and does the gap grow with k (Table 1, Figure 1c)? What is a better verifier worth against one more draw (Table 2)? Which axis of the design carries the gain (Table 5, Figure 3 right)? What does more budget buy, and for whom (Figure 1d, Figure 7)? And does any of it replicate on another release (Table 8)? Throughout, PATIENCE is PATIENCE, "AVA+menu" is AVA's rule with all four operations, and every number is at the cell's budget unless a sweep is named.

Operations beat samples, and the gap is larger at k=4. Table 1 gives pass^4 for every rule in every cell, and Figure 3 (appendix) shows the same numbers as gains over base, largest in the rows with the most room. The cell values are exact for the recorded pools and carry no sampling interval, while the contrasts in Table 4 (appendix) carry intervals because they generalize over tasks. Every allocator sits above the base agent in the mean row. 5 of the seven rules are above it in all twelve cells, and PATIENCE is above it in eleven, from 5.1 points on gpt-4.1-mini retail to -0.5 on o4-mini airline. Table 4 (appendix) puts intervals on this. Over base, PATIENCE gains +0.71 points at k=1, +1.76 at k=4, and +1.80 at k=8, and every interval excludes zero. Figure 1c shows that every rule's advantage over base is larger at k=4 than at k=1, which is what Proposition 1 predicts and the reason to measure pass^k rather than accuracy when comparing allocators; at k=1 the same methods look almost identical. From k=4 to k=8 the rules that vote hold or widen their advantage while the verifier-gated rules, those whose stopping depends on the verifier's score, slip slightly. Table 16 breaks the main contrast out by domain, +3.03 points on retail, +1.01 on telecom, and +0.60 on airline, the one domain, at fifty tasks, whose interval reaches zero.

Table 2: One more draw against a perfect verifier, retail, mean of four models, pass^4 points.

Rule (retail, mean of 4 models)	pass^4 at $3\times$	one more draw ($3\times \to 4\times$)	perfect verifier at $3\times$
AVA	57.2	+0.6	+2.1
AVA+menu	58.1	+0.4	+2.3
PATIENCE	59.5	+1.6	+0.5

A draw versus a perfect verifier. Table 14 (appendix) runs the rules under three verifiers up to the label itself, which moves PATIENCE from 42.9 to 44.2 mean pass^4. Table 2 sets that against one more draw. For PATIENCE, raising the retail budget from $3\times$ to $4\times$ the median transcript is worth +1.6 points and a perfect verifier at $3\times$ is worth +0.5; for AVA+menu the ranking reverses, +0.4 against +2.3. Two published results bound the verifier side. For a rule that accepts the first candidate the verifier passes, the large-budget success probability is $p\,\mathrm{TPR}/(p\,\mathrm{TPR} + (1-p)\,\mathrm{FPR})$ (Stroebl et al., 2026), and the verifier's ROC curve fixes best-of-N accuracy at every budget (Dorner et al., 2026). Our verifier's out-of-fold rates at the cascade's threshold are TPR 0.68 and FPR 0.49 on retail and 0.55 and 0.08 on telecom. At those rates the ceiling is 50.8% pass^4 over the twelve cells. The cascade reaches 42.7% under the byte budget, so the budget binds well before the verifier's own limit. A rule built from it, drawing while a final verification stays affordable and then committing the verified-best, reaches 43.6% mean pass^4 with the label as verifier. The same draw-first rule with a vote in place of the verifier reaches 43.5%, and best-of-k, which reserves bytes for its vote and so draws slightly differently, reaches 43.3%. The perfect verifier is no better than either, because a verifier that must read every candidate costs as much as the candidates and cuts the draws in half. Make that verification free and the rule draws 2.3 times per run and reaches 53.2% (70.2% on retail), against the idealized three-draw ceiling of 59.4%. Selection accuracy is worth about a point here (42.9% to 44.2% for PATIENCE) and selection cost about ten. That is the cost-over-accuracy rule; the vote wins because it is nearly free, not because it is good. Charging the perfect verifier half, a quarter, or a tenth of the bytes it reads makes it worth 46.5%, 49.7%, and 51.5% (Table 12, appendix), so each halving of the read cost buys about three points. The cheap rules' two points are 13% to 16% of the 11.8 points free perfect selection would open at this budget; the rest is the price of selection.

The rule is not the accounting. Charging the verifier for its reads does not decide the result by itself. It fixes how many draws the verifier leaves, so a perfect verifier that reads everything is worth Theorem 1's ceiling at those draws. Where the vote lands is a separate fact. At this budget the vote makes 2.3 draws on average, perfect selection over those same draws would reach 52.5%, and the vote reaches 43.3%, so it captures 17% of that gain (4% to 29% across cells, Table 10, appendix). Tying the paid perfect verifier at this budget takes 20%. Proposition 2 predicts the curve of Table 12 from the pools with draws rounded down (Table 13, appendix). It lands within 0.8 points of the exact rule at read costs of 50%, 25%, and 10% and misses by up to 3.0 points at the two ends, where rounding decides whether a second candidate fits. It puts the tie with the vote at a read cost of 0.68; the observed tie is near 1.

The verifier the rule recommends. We built one. It reads only a candidate's canonical write-action set, 261 bytes on average against a 12,210-byte transcript, as a logistic model over the set's action tokens fitted out of fold by task, so it never sees a run of the task it scores. Its AUC is 0.65, 0.70, and 0.93 by domain, about the text verifier's, at 2% of the read cost. Through the draw-first rule it reaches 47.7% mean pass^4 (Table 11, appendix), 4.2 points above the draw-first vote and 4.1 above the perfect verifier that reads whole transcripts, at the same bytes. It wins in every domain (61.7 against 59.4 on retail, 45.8 against 41.8 on airline, 35.5 against 29.3 on telecom), with selection efficiency 62% against the vote's 17%; the verifier-weighted and filtered votes of the best-of-N literature, run with the transcript verifier, trail the plain vote for the same reason that verifier does (Table 9, appendix). This is what Proposition 2 says a selector at near-zero read cost with efficiency well above the vote's should do, and Figure 6 (appendix) places every selector in the study on the boundary the proposition draws. Figure 1d and 1e show the rule on retail and as four bars.

The menu is the lever. Two contrasts isolate it. Giving AVA's rule the JUDGE and RETRY operations adds +0.84 points of pass^4 [+0.47, +1.27]; giving PATIENCE the same two operations over its version restricted to sample and verify adds +0.76 [+0.21, +1.31]. The menu effect holds in every domain for the threshold rule (Table 16). Figure 1d says why. On retail the base agent sits at 56.5%; sample-and-verify climbs to 57.2% at $3\times$ and saturates; handing that rule a perfect verifier that is paid for

lifts it only to 59.0%, a gap of 1.8 points for all the accuracy a verifier could ever add; making the same perfect selection free lifts it to 70.2%, a gap of 11.3 points for the cost alone; and Theorem 1's three-draw ceiling sits at 76.3%. On retail, the sample-and-verify rule rises from 56.7% at $1.5\times$ to 58.6% at $6\times$ the median transcript, about two points above base, while PATIENCE rises from 57.7% to 62.3% and best-of-k to 62.5%, about six points above base. The verifier is weak on retail (AUC 0.63) and cannot convert more budget into reliability; a vote can, because the passing runs of a sometimes-task tend to agree on their writes while the failing runs disagree. On telecom the verifier is strong (AUC 0.88) and Table 1 flips the ordering, with AVA+menu ahead of best-of-k; which operation is worth buying depends on which signal is good in the domain.

More budget buys consistency, but only through the vote. Figure 7a (appendix) shows PATIENCE's pass^k surface over budget and k on retail against the base agent's plane. It rises with budget at every k, and the gap is largest at $k{=}8$, where the base agent has decayed furthest. Table 3 (appendix) gives the sweep for six rules with the Theorem 1 ceiling. Best-of-k and PATIENCE are within a fraction of a point at every budget, and PATIENCE spends fewer bytes at every budget up to $4\times$ and slightly more at $6\times$. AVA and AVA+menu flatten from $3\times$ onward because their stopping rule exits once the verifier is confident, which on retail it is too often. Task by task (Figure 7b), PATIENCE raises the commit probability on 17% of the 1,112 pairs and lowers it on 12%; tasks at 0 or 1 cannot move. Budget buys consistency, but only for the rules that vote.

The objective did not matter. Two expectations were written down before any result was computed, that tuning for consistency would change the allocation and that a learned controller would beat fixed rules; the data refuted both. Tuning the threshold rule to maximize pass^4 instead of accuracy selected the same two thresholds in all twelve cells, so the two AVA+menu rows of Table 5 are identical. Training PATIENCE on pass^k targets instead of accuracy targets moved pass^4 by -0.01 points [-0.49, +0.46], and the null holds in every domain (Table 16). The condition in Section 2 says when the objectives separate, namely when one operation targets near-certain tasks and another targets hard ones. On this menu every operation lifts the same population, the sometimes-tasks, so the objectives agree; that is a fact about the menu, not the metric.

The effects replicate on a second release. Table 8 (appendix) repeats the study on the original τ-bench trajectories, a different benchmark version, user simulator, and model generation. There, as on the newer release, the rules that vote widen their gain over base from $k{=}4$ to $k{=}8$ while the verifier-gated rules hold. Over 330 pairs PATIENCE gains +1.98 points of pass^4 over base [+1.05, +2.98], best-of-k gains +2.83, and AVA's menu effect is +1.19 [+0.51, +1.96]; retail gains +3.19 and airline -0.80, so airline is the weak domain on both releases.

Learning did not beat a tuned rule or a vote. PATIENCE is within noise of AVA+menu (-0.06 [-0.64, +0.51]) and of best-of-k (-0.18 [-0.67, +0.29]), leading on retail and trailing on telecom (Table 1). An imitation variant that copies the action the dynamic program marks best reached 57.0 on retail against 59.5 for the value heads. What it learned transfers. trained on one domain and applied to another without retraining, it matches the in-domain controller over all six source-target pairs (+0.29 points [-0.23, +0.79], Table 15). We read this as data scale, since PATIENCE has at most 114 tasks per domain to learn from. What it learned is legible. Figure 5 (appendix) shows that it samples while the budget is fresh, mixes verification and voting in the middle, and votes when the budget is nearly gone, the order a person would choose once one notices that a vote is cheap and a sample is not.

Spend, in bytes and dollars. Every rule ran under the same byte meter, and PATIENCE matches best-of-k with about twelve % fewer bytes, the one place learning paid, in cost rather than reliability. At the prices Sierra recorded for the agent's own calls, PATIENCE's extra 13.2 KB per run costs between $0.020 and $0.39 per conversation for its +1.76 points of pass^4 (appendix). The operational reading is to buy candidates and a vote unless a telecom-grade verifier is available, and then to gate on it.

Limitations. Three things the numbers do not show. Pools of four or eight trajectories cap the diversity a live sampler would find. The recorded agents ran at temperature zero, so a SAMPLE buys a different simulated user rather than a different decode. That is the benchmark's own source

of trial-to-trial variation, but not the only one a deployment has. A vote can also commit a wrong action that several runs agree on, the correlated-jury failure of Ladha (1992). In 18% of the 617 sometimes-tasks the plurality write set is wrong although a passing run is in the pool. A split-half check on the eight-trial pools (Table 6, appendix), predicting from four trials and scoring on the other four, puts population predictions within 2.8 points at $k{=}1$ and 7.7 at $k{=}4$ with the vote's advantage replicating in direction every time; a live check at positive temperature is next.

6 Related Work

Allocating test-time compute. Snell *et al.* (Snell et al., 2025) allocate compute per prompt by predicted difficulty; Wu *et al.* (Wu et al., 2025) compare voting, best-of-n, and tree search at matched FLOPs from cached samples; Brown *et al.* (Brown et al., 2024) characterize coverage under repeated sampling. Chen *et al.* (Chen et al., 2024a) show voting can be non-monotone in the number of calls; FrugalGPT (Chen et al., 2024b) cascades models under a dollar budget. Among recent work, CaTS (Huang et al., 2026) stops best-of-N early with a calibrated confidence, and Best-of-Infinity (Komiyama et al., 2026) studies ensembling as N grows. Three papers speak to the verifier itself. ROC-n-reroll (Dorner et al., 2026) ties best-of-N accuracy to the verifier's ROC curve, Stroebl et al. (2026) show that a verifier's false positives cap resampling at any budget, and Setlur et al. (2025) prove that verifier-based scaling beats verifier-free scaling asymptotically, without charging for verification. Our headline is what happens to the verifier's advantage once its reads are paid for.

Agents, budgets, and consistency. τ-bench (Yao et al., 2025) introduced pass^k; τ^2-bench (Barres et al., 2025) kept it as its primary metric and released the trajectories we replay; Hochlehnert et al. (2025) document the seed sensitivity that motivates exact replay. AVA (Patel, 2026) is the nearest work and our reimplemented baseline; the differences are the two extra operations, the target, the byte unit, and the agentic setting. ARPO (Dong et al., 2026) branches rollouts adaptively under a tool-call budget and reports pass@5, any success among five, the complement of what pass^k counts; Fang *et al.* (Fang et al., 2026) and Dip *et al.* (Dip et al., 2026) allocate under token budgets. None has a vote or a retry, and none targets pass^k.

Verifying agents. Self-consistency (Wang et al., 2023) votes over final answers; JUDGE votes over write-action sets. Tool-call reward models (Ma et al., 2026), trajectory judges (Zhuge et al., 2025; Xia et al., 2025), and step-level verifiers (Lightman et al., 2024; Cobbe et al., 2021) are stronger signals than ours.

Sequential allocation. The buying problem is a bandit with knapsacks (Badanidiyuru et al., 2013; 2018; Tran-Thanh et al., 2010; 2012; Whittle, 1988); agreement stopping descends from Gittins indices and Pandora's box, and its committed variant (Gittins, 1979; Weitzman, 1979; Doval, 2018); the ratio rule is adaptive-submodular greedy with the knapsack guarantee of Khuller et al. (1999); Sviridenko (2004); learning from logs follows Swaminathan & Joachims (2015); background in Slivkins (2019); Lattimore & Szepesvári (2020).

7 Conclusion

The question was which operation the next byte should buy, and the measured answer inverts the usual priority. It should buy something other than one more sample, and what matters at deployment budgets is not how accurately a candidate is selected but how cheaply. A perfect verifier that must read the candidates it judges is worth about a point over a cheap one, because it costs what a candidate costs and cuts the draws in half; the same selection made free is worth ten. That is the cost-over-accuracy rule, and Proposition 2 says where it stops holding. The vote wins today because it is nearly free, not because it is good. Operations beat samples at equal bytes, the menu is the lever, tuning for consistency changed no decision, and a learned controller matched a tuned rule. The signal worth building is a cheap one; the first we built, reading only write actions, beats the vote by four points.

AI USE STATEMENT

We used a large language model as an assistant for drafting and editing the text, for drafting the TikZ and PGFPlots figure code, the trajectory extraction script, the exact replay evaluator, the allocation rules, the verifiers, and the analysis scripts from our own data and specifications, for running those scripts on the public trajectory releases, for drafting the statements and proofs of the propositions and the theorem. The research question, the choice of setting, metric, and operations, and the design of the study are the authors' own. No synthetic data were generated. Every reported number is recomputed by the released scripts from the released trajectories, and the authors reviewed all AI-assisted text, code, and proofs. We take full responsibility for the content of this paper.

ETHICS STATEMENT

The study replays public benchmark trajectories of simulated customers and introduces no new capability. Its practical message is that a byte budget is better spent on candidates and a cheap vote than on an expensive verifier; an operator applying that in a high-stakes domain should remember that a vote can commit a wrong action several runs agree on, and that a domain with a strong verifier changes the answer. No human subjects, private data, or new datasets are involved.

REPRODUCIBILITY STATEMENT

The evaluator, the extraction script, the verifier and controller code, and the figure and table generators are released as anonymized supplementary material. The input data are the trajectory files shipped in the public τ^2-bench and τ-bench repositories; the extraction script reproduces our flat dataset from them, and our re-scoring reproduces both benchmarks' published pass^1 values. Every number in this paper, including those in the prose, is emitted by the release code from the results files; nothing is entered by hand. Folds, seeds, budgets, threshold grids, and verifier definitions are stated in Sections 3 and 4 and fixed in the code. A full run of all sixteen cells, the budget sweep, the three verifiers, the transfer pairs, and the split-half validation takes about an hour on one laptop core with no accelerator; no model is queried at any point, so the study has no API cost and no dependence on provider state at the time of reading.

REFERENCES

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 207–216, 2013. doi: 10.1109/FOCS.2013.30.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *Journal of the ACM*, 65(3):13:1–13:55, 2018. doi: 10.1145/3164539.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. arXiv:2506.07982.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. arXiv:2407.21787.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024a. arXiv:2403.02419.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024b. OpenReview cSimKw5p6R; arXiv:2305.05176.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan,

Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. arXiv:2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. arXiv:2110.14168.

Sajib Acharjee Dip, Dawei Zhou, and Liqing Zhang. Think again or think longer? selective verification for budget-aware reasoning, 2026. arXiv:2606.19808.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.19849.

Florian E. Dorner, Yatong Chen, André F. Cruz, and Fanny Yang. ROC-n-reroll: How verifier imperfection affects test-time scaling. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.12399.

Laura Doval. Whether or not to open Pandora's box. *Journal of Economic Theory*, 175:127–158, 2018. doi: 10.1016/j.jet.2018.01.005.

Zhengru Fang, Senkang Forest Hu, Zhonghao Chang, Yu Guo, Yihang Tao, Hongyao Liu, Mengzhe Ruan, Jun Huang, and Yuguang Fang. Inference-time budget control for LLM search agents, 2026. arXiv:2605.05701.

John C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B*, 41(2):148–164, 1979. doi: 10.1111/j.2517-6161.1979.tb01068.x.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011. doi: 10.1613/jair.3278.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. In *Proceedings of the Second Conference on Language Modeling (COLM)*, 2025. doi: 10.48550/arXiv.2504.07086. OpenReview 90UrTTxp50; arXiv:2504.07086.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. CaTS: Calibrated test-time scaling for efficient LLM reasoning. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026. OpenReview jrSc4RJXy1; arXiv:2503.00031.

Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999. doi: 10.1016/S0020-0190(99)00031-9.

Junpei Komiyama, Daisuke Oba, and Masafumi Oyamada. Best-of-infinity: Asymptotic performance of test-time LLM ensembling. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.21091.

Krishna K. Ladha. The Condorcet jury theorem, free speech, and correlated votes. *American Journal of Political Science*, 36(3):617–634, 1992.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. doi: 10.1017/9781108571401.

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 *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. OpenReview v8L0pN6EOi.

Da Ma, Ziyue Yang, Hongshen Xu, Haotian Fang, Kai Yu, and Lu Chen. Empowering LLM tool invocation with tool-call reward model. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026. OpenReview LnBEASInVr.

Dipkumar Patel. Anytime verified agents: Adaptive compute allocation for reliable LLM reasoning under budget constraints. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. OpenReview JMDCMf7mlF.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without verification or RL is suboptimal. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54058–54094, 2025. arXiv:2502.12118.

Aleksandrs Slivkins. Introduction to multi-armed bandits, 2019. arXiv:1904.07272.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. arXiv:2408.03314.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. The limits of inference scaling through resampling. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026. arXiv:2411.17501.

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004. doi: 10.1016/S0167-6377(03)00062-2.

Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *Proceedings of the 32nd International Conference on Machine Learning (ICML), PMLR 37*, pp. 814–823, 2015.

Long Tran-Thanh, Archie Chapman, Enrique Munoz de Cote, Alex Rogers, and Nicholas R. Jennings. Epsilon-first policies for budget-limited multi-armed bandits. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 1211–1216, 2010. doi: 10.1609/aaai.v24i1.7758.

Long Tran-Thanh, Archie Chapman, Alex Rogers, and Nicholas R. Jennings. Knapsack based optimal policies for budget-limited multi-armed bandits. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1134–1140, 2012.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. arXiv:2203.11171.

Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979. doi: 10.2307/1910412.

Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25A:287–298, 1988.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. arXiv:2408.00724.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentRM: Enhancing agent generalization with reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19277–19290, 2025. doi: 10.18653/v1/2025.acl-long.945.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.12045.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning (ICML), PMLR 267*, pp. 80569–80611, 2025. arXiv:2410.10934.

A Proposition 1, general form

Let tasks have success probabilities p_τ and let an operation o map p_τ to $p_\tau^o \in [p_\tau, 1]$. The accuracy gain of o is $\mathbb{E}[p^o - p]$ and its pass^k gain is $\mathbb{E}[(p^o)^k - p^k]$. For any pair in which o_2 moves a set of tasks to exactly 1 while o_1 moves a disjoint set from $p < 1$ to $p' < 1$, the pass^k gain of o_1 tends to zero as $k \to \infty$ while that of o_2 tends to the probability of its set, so for large enough k, o_2 maximizes pass^k whenever o_1 maximizes accuracy. The worked numbers in the main text show the crossing already at $k = 2$.

B Verifier and judge details

Features per trajectory, all observable before grading. Tool-error count and a flag for any error; number of assistant turns; number of tool calls; number of write actions and a flag for none; transfer-to-human flag; transcript kilobytes; one-hot termination reason; one-hot model. Logistic regression, $C = 1$, standardized features, five task-grouped folds per domain. Write tools per domain are the functions typed as write operations in the τ^2-bench source (seven each in retail and airline, six in telecom, plus transfer to human). A write-action set is the sorted list of canonical call strings; two runs agree when their sets are identical.

C Reimplemented AVA and tuned thresholds

Grid $\tau \in \{0.3, 0.6\}$, $\theta \in \{0.7, 0.85, 0.95\}$, tuned in sample per cell to maximize the target; the selected pair is stored in every results file. The verifier-weighted vote weights each candidate's write set by its verifier score; the verifier filter keeps the candidates scored at least 0.5 and falls back to all candidates when none is; the difficulty-gated rule verifies the first candidate and commits when its score is at least 0.85. The rule verifies an unscored pool before drawing, exits when confidence reaches θ, draws only while uncertainty exceeds τ and the budget allows, and commits the verified-best candidate; with the extended menu it retries an errored commit once and votes before committing when two or more candidates exist.

D Learned controller details

Histogram gradient-boosted regression per operation, depth 4, 100 rounds, learning rate 0.1, minimum 15 samples per leaf, standardized inputs, at most 400 training states per task sampled with a fixed seed. Targets are exact $Q^*(s, o) - V_{\text{commit}}(s)$ from dynamic programming under the cell's budget with at most five draws and one retry per run. Five task-grouped folds shared across the four models of a domain. The imitation variant fits a gradient-boosted classifier over the action the dynamic program marks best, including commit, on the same states.

E Design

Figure 2 is the full design behind Figure 1a, from the recorded runs through the operations and the controller to the exact evaluation.

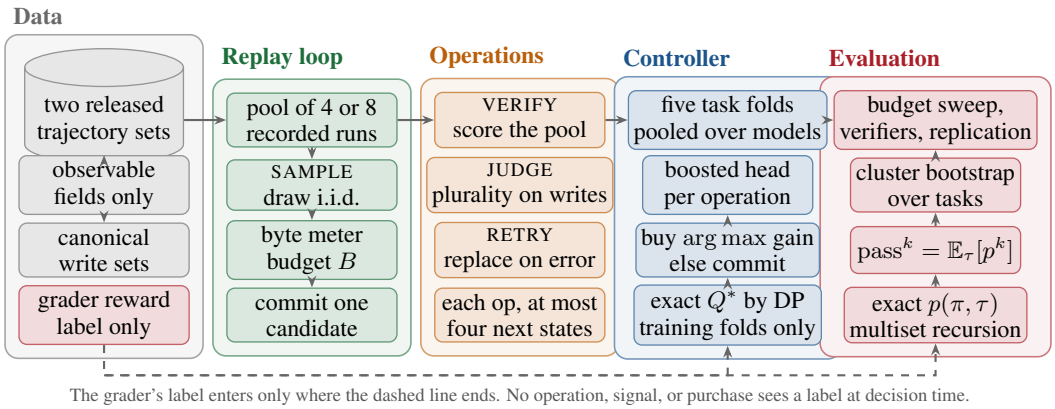

Figure 2: The study in five layers; the dashed line marks the only two places the label enters.

What to notice is where the label enters. It is used at two points only, to score a committed candidate and to fit the value targets on other folds; nothing between the pool and the commit sees it, which is what makes the comparison of rules fair.

F BUDGET SWEEP AND OPERATION MIX

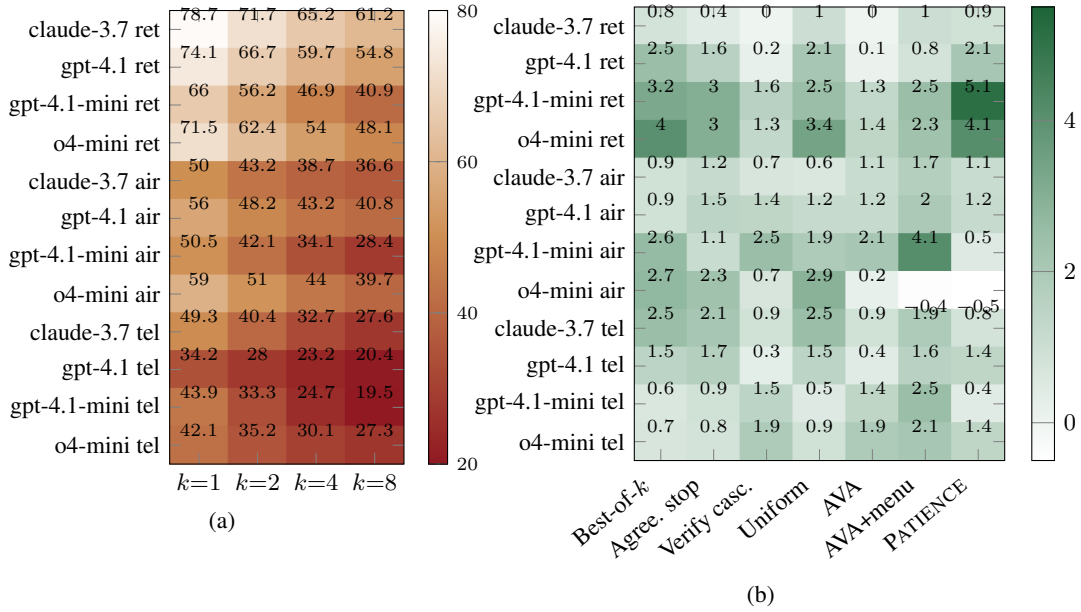

Figure 3: pass^k against k for the released agents in panel (a) and gain over the base agent at equal byte spend in panel (b), twelve cells, one per model and domain, of the released τ^2-bench runs.

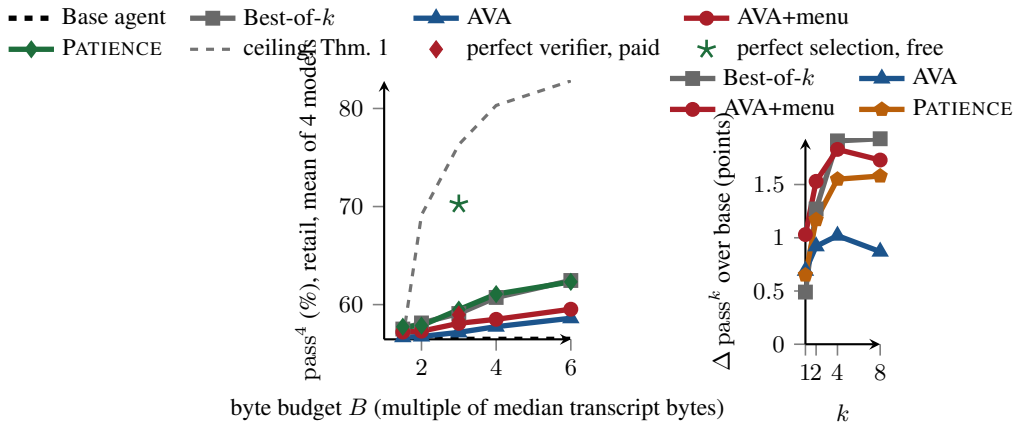

Figure 4: Panel (a), pass^4 against the byte budget on retail, mean of four models, with the two perfect-selection points at 3× and the ceiling of Theorem 1; panel (b), gain over the base agent against k, mean of twelve cells.

Table 3 gives the retail sweep behind Figure 1d, and Figure 4 is its large version with the bytes each rule spent, and Figure 5 shows what PATIENCE buys as its budget runs down.

Table 3: Budget sweep on retail, mean of four models, pass^4 (%), with the Theorem 1 ceiling; best allocator per row in bold.

Budget B	draws m	Base	Best-of-k	Agree. stop	AVA	AVA+menu	PATIENCE	ceiling, Thm. 1
1.5×	1	56.5	57.5	57.4	56.7	57.2	**57.7**	56.5
2×	2	56.5	58.1	**58.1**	56.7	57.3	57.9	69.1
3×	3	56.5	59.1	58.5	57.2	58.1	**59.5**	76.3
4×	4	56.5	60.7	58.6	57.7	58.5	**61.1**	80.3
6×	5	56.5	**62.5**	58.7	58.6	59.5	62.3	82.8

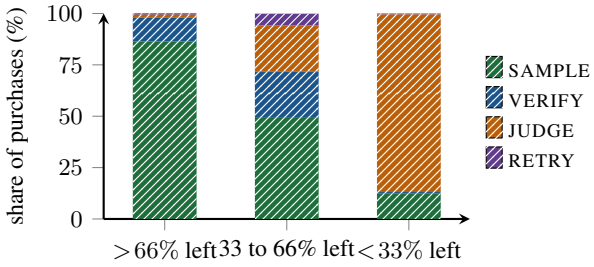

Figure 5: Operation mix of PATIENCE by remaining budget, twelve cells.

Three things stand out. At every budget the rules that vote keep gaining while the verifier-gated rules flatten, so more bytes buy consistency only through more candidates. The ceiling column shows how far all of them sit from perfect selection, 76.3% at 3× against 62.5% for the best rule at 6×, so the budget never binds for a perfect selector. And Figure 5 shows PATIENCE buying samples while the budget is fresh and votes when it is nearly gone, which is the order the cost-over-accuracy rule recommends.

G PAIRED CONTRASTS

Table 4 gives every paired contrast in the study with its cluster-bootstrap interval; Figure 1f plots five of them.

Table 4: Paired contrasts in points of pass^k, cluster bootstrap over tasks; * excludes zero.

Contrast (points of pass^k)	k=1	k=4	k=8
PATIENCE minus base agent	+0.71 [+0.29, +1.11]*	+1.76 [+1.22, +2.34]*	+1.80 [+1.29, +2.34]*
Best-of-k minus base agent	+0.49 [+0.03, +0.92]*	+1.94 [+1.39, +2.48]*	+1.98 [+1.46, +2.49]*
AVA+menu minus base agent	+0.93 [+0.57, +1.32]*	+1.83 [+1.29, +2.42]*	+1.73 [+1.22, +2.33]*
AVA+menu minus AVA (menu effect)	+0.29 [+0.02, +0.56]*	+0.84 [+0.47, +1.27]*	+0.88 [+0.50, +1.31]*
PATIENCE minus PATIENCE (S, V) (menu effect)	+0.18 [-0.23, +0.58]	+0.76 [+0.21, +1.31]*	+0.78 [+0.27, +1.31]*
PATIENCE minus accuracy-target PATIENCE (target effect)	+0.06 [-0.27, +0.38]	-0.01 [-0.49, +0.46]	+0.08 [-0.38, +0.54]
PATIENCE minus AVA+menu (rule effect)	-0.23 [-0.64, +0.18]	-0.06 [-0.64, +0.51]	+0.07 [-0.49, +0.60]
PATIENCE minus best-of-k	+0.22 [-0.16, +0.61]	-0.18 [-0.67, +0.29]	-0.18 [-0.64, +0.28]

Six of the nine contrasts exclude zero and all six survive a Holm correction; the two that do not are the target effect and the rule effect, the paper's two negative results. The takeaway is that the effects a reviewer would expect to be large, the objective and the learner, are the ones that vanish, while the cheap ones, another candidate and a vote, are the ones that hold.

H FACTORIAL DESIGN

Table 5 lays out the rule, menu, and target factors whose paired contrasts appear in Table 4.

Table 5: Factorial over rule, menu, and target, mean of twelve cells.

Rule	Menu	Target	pass^1	pass^4	pass^8
Base agent	one sample	none	56.3	41.4	37.1
Threshold (AVA rule)	S, V	accuracy	57.0	42.4	38.0
Threshold (AVA rule)	S, V, J, R	accuracy	57.3	43.2	38.8
Threshold (AVA rule)	S, V, J, R	pass^k	57.3	43.2	38.8
Learned values	S, V	pass^k	56.8	42.3	38.0
Learned values	S, V, J, R	accuracy	57.1	43.2	38.9
Learned values	S, V, J, R	pass^k	56.9	42.9	38.7

Reading down the rows, adding JUDGE and RETRY to the menu moves pass^4 from 42.3 to 42.9, while changing the training target from accuracy to pass^k moves it from 43.2 to 42.9, a difference inside the design's resolution. The menu is the lever and the target is not, which is what Table 4 says with intervals.

I SPLIT-HALF VALIDATION OF THE EVALUATOR

Table 6 scores predictions made from four recorded trials against the other four, held out, on the eight-trial pools; it bounds the plug-in bias discussed in Section 3.

Table 6: Split-half validation, predictions from four trials against the held-out four, claude-3.5-sonnet on the original τ-bench release.

Model	Domain	Rule	k	predicted	held out	MAE per task	r
claude-3.5	airline	Base agent	k=1	45.5	46.5	16.0	0.78
claude-3.5	airline	Base agent	k=4	26.6	30.2	15.1	0.68
claude-3.5	airline	Best-of-3 vote	k=1	46.4	47.3	17.4	0.78
claude-3.5	airline	Best-of-3 vote	k=4	30.4	34.0	18.2	0.66
claude-3.5	retail	Base agent	k=1	70.7	67.8	17.6	0.72
claude-3.5	retail	Base agent	k=4	57.6	50.0	22.2	0.62
claude-3.5	retail	Best-of-3 vote	k=1	72.0	72.8	16.4	0.72
claude-3.5	retail	Best-of-3 vote	k=4	59.9	61.8	21.4	0.61

The largest error is the base agent on retail at k=4, where four trials overpredict the held-out four by 7.7 points; that is the plug-in bias of Section 3, and it is the same for every rule. In every row the vote's advantage over the base agent keeps its sign on the held-out trials. The takeaway is that the

evaluator's population numbers carry a bias that grows with k, while its comparisons between rules do not.

J EVALUATION DESIGNS OF THE CLOSEST STUDIES

Table 7: Evaluation designs of the closest prior studies and of this one.

Study	Task type	Budget unit	Target	Data and models
Snell et al. (2025)	single-turn math	generations, FLOPs	accuracy	MATH; one base model family
Wu et al. (2025)	single-turn math	inference FLOPs	accuracy	MATH, GSM8K; Llemma, Mistral, Pythia
Chen et al. (2024a)	QA and fact checking	number of LM calls	accuracy	four datasets; GPT-3.5
Huang et al. (2026)	single-turn reasoning	samples (16)	accuracy	nine datasets; three 1.5B to 8B models
Dorner et al. (2026)	math and code	samples	accuracy	four datasets; Qwen3 and Llama-3 families
Patel (2026)	reasoning, QA, code	tokens, tool calls	reliability, accuracy	GSM8K, MATH, HotpotQA, HumanEval; GPT-5, GPT-4o
This paper	multi-turn tool use	transcript bytes	pass^k	2 releases, 3 domains, 6 models, 6,428 runs; exact replay

The table is read by row. Each earlier study fixes one kind of spending, budgets in FLOPs, samples, or tokens, and targets accuracy on single-turn problems; the last row is the only multi-turn, byte-metered, consistency-targeted design, and the only one scored without sampling. That is the gap the paper fills, and the reason none of the earlier results can be read off for agents.

K REPLICATION ON THE ORIGINAL RELEASE

The k=8 column appears only for the eight-trial model, where it is exact.

Table 8: Replication on the original τ-bench release, pass^k (%) at equal byte spend; best allocator per row in bold.

Model	Domain	k	Base	Best-of-k	Agree. stop	Verify casc.	Uniform	AVA	AVA+menu	PATIENCE
gpt-4o (4 trials)	retail	k=4	44.9	**48.1**	47.3	46.2	47.8	45.8	47.8	47.5
claude-3.5-sonnet (8 trials)	retail	k=4	49.7	**53.9**	52.0	50.2	53.6	50.3	51.8	53.4
claude-3.5-sonnet (8 trials)	retail	k=8	42.2	**47.2**	45.0	42.8	46.8	42.9	44.4	46.6
gpt-4o (4 trials)	airline	k=4	23.9	24.1	23.4	**24.5**	23.2	24.2	24.3	22.7
claude-3.5-sonnet (8 trials)	airline	k=4	25.3	26.6	25.6	**27.0**	26.6	26.9	26.8	24.9
claude-3.5-sonnet (8 trials)	airline	k=8	19.6	21.0	20.2	**21.5**	20.9	21.4	21.2	19.1

Both of the paper's main effects replicate on a release with different models, a different user simulator, and a different year: operations beat samples, and the rules that vote widen their gain from k=4 to k=8 while the verifier-gated rules hold. Airline is again the weak domain, with fifty tasks and low base rates. The takeaway is that the findings are about the setting, not about one release.

L RULES THAT USE THE VERIFIER INSIDE THE VOTE

Table 9 scores the verifier-weighted vote, the verifier filter, and the difficulty-gated rule discussed in Section 5, under the transcript-statistics verifier and under the oracle. Three rules from the best-of-N literature use the verifier inside the vote rather than instead of it, a verifier-weighted plurality, a verifier filter followed by a plurality (the reranker-against-filter distinction of Dorner et al. (2026)), and a difficulty-gated rule in the spirit of Snell et al. (2025) and CaTS (Huang et al., 2026). Table 9 scores them. With the transcript-statistics verifier all three trail the plain vote (42.2, 42.3, and 41.9 against 43.3), because each pays to read the candidates and so draws fewer. With the label as verifier, the weighted vote and the filter collapse to the verified-best rule at 43.6, still no better than the plain vote. Where the verifier sits inside the rule does not matter here; that it is paid for does.

Table 9: Rules that use the verifier inside the vote, twelve cells, pass^4 (%) and kilobytes spent.

Rule	pass^4, statistics verifier	KB spent	pass^4, oracle verifier
Best-of-k, plain vote	43.3	29.1	
Verifier-weighted vote	42.2	25.2	43.6
Verifier filter, then vote	42.3	25.2	43.6
Difficulty-gated (commit if first score ≥ 0.85)	41.9	24.2	42.3
Draw first, verified-best	42.8	25.2	43.6

M SELECTION EFFICIENCY OF THE VOTE

Table 10 gives, per cell, the base agent, the plurality vote at its actual draws, Theorem 1's ceiling at those same draws, the share of that gain the vote captures, the paid perfect verifier, and the share the vote would need to tie it.

Table 10: The vote's selection efficiency against the same-draw ceiling, twelve cells, pass^4 (%).

Model	Domain	Base	Vote	Ceiling, same draws	Vote efficiency	Perfect verifier, paid	Efficiency to tie
claude-3-7-sonnet	airline	38.7	39.6	44.9	14%	41.4	43%
claude-3-7-sonnet	retail	65.2	66.0	75.5	8%	66.8	16%
claude-3-7-sonnet	telecom	32.7	35.2	45.0	20%	33.2	4%
gpt-4.1	airline	43.2	44.1	50.1	13%	44.5	18%
gpt-4.1	retail	59.7	62.2	71.2	22%	61.0	11%
gpt-4.1	telecom	23.2	24.7	30.4	21%	23.5	4%
gpt-4.1-mini	airline	34.1	36.7	47.4	19%	39.0	36%
gpt-4.1-mini	retail	46.9	50.1	62.0	21%	50.6	24%
gpt-4.1-mini	telecom	24.7	25.3	38.0	4%	27.7	23%
o4-mini	airline	44.0	46.7	55.1	25%	44.9	8%
o4-mini	retail	54.0	58.0	67.8	29%	57.5	25%
o4-mini	telecom	30.1	30.8	43.0	5%	33.0	22%

The vote captures between 4% and 29% of the gain a perfect selector over the same draws would, and the share it needs to tie the paid perfect verifier runs from 4% on gpt-4.1 telecom, where the paid verifier barely beats the base agent, to 43% on claude-3-7-sonnet airline. The tie in the mean row is therefore a fact about averages; cell by cell either side can win. The takeaway is that a better cheap selector has most of the same-draw ceiling still to claim.

N WHERE VERIFYING BEATS VOTING

Figure 6 places the study's selectors in the plane of read cost and selection efficiency. The solid boundary is measured, the paid perfect verifier's value at the five read costs of Table 12 expressed as the share of the free-selection gain a cheap selector would need to tie it; the dashed boundary is Proposition 2's closed form with divisible draws, which overstates the share because the exact rule cannot draw a fraction of a candidate. A selector above the boundary at its read cost beats the paid perfect verifier; the vote sits below it at full read cost, the write-set verifier far above it at a read cost near zero, and the transcript verifier, at 57% efficiency once its reads are free, is sunk by the reads alone.

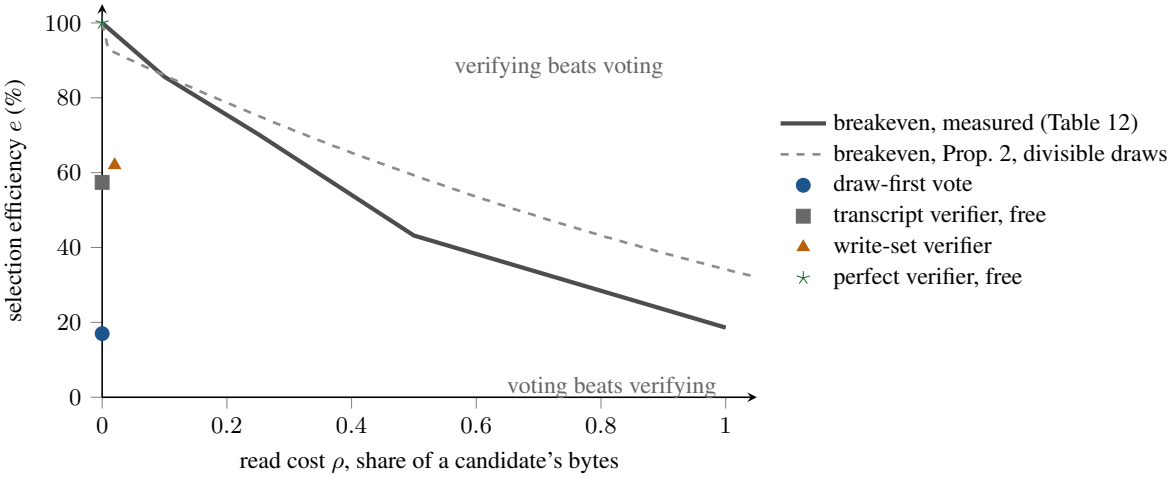

Figure 6: Selection efficiency against read cost, with the measured and predicted breakeven boundaries and the four selectors of the study.

O THE WRITE-SET VERIFIER

Table 11 runs the rules with the verifier that reads only the write-action set, charged only those bytes plus the fixed overhead; its AUC by domain is 0.65, 0.70, and 0.93.

Table 11: Rules under the write-set verifier, twelve cells, pass^4 (%) and kilobytes spent.

Rule	pass^4, write-set verifier	KB spent
Best-of-k, plain vote	43.3	29.1
Draw first, vote	43.5	28.3
Draw first, verified-best	47.7	28.0
Verify cascade	46.3	23.2
Verifier-weighted vote	44.0	19.3
Verifier filter, then vote	44.4	19.3

The rule that gains most is draw-first verified-best, 47.7% at 28.0 KB, and the verify cascade follows at 46.3%. Used inside the vote as a weight or a filter the verifier reaches 44.0% and 44.4% while spending 19.3 KB, a third less than best-of-k. The takeaway is that a selector that is both cheap to read and better than a vote pays twice, in pass^4 and in bytes.

P VERIFIER READ COST

Table 12 charges the draw-first rule's perfect verifier a decreasing share of the bytes it reads, from the full cost used everywhere else in the paper down to none, and reports what the rule reaches against the plain vote.

Table 12: The perfect verifier's value as its read cost falls, twelve cells, pass^4 (%).

Share of candidate bytes the verifier is charged	pass^4, draw-first with the label as verifier	against the plain vote
100%	43.6	+0.1
50%	46.5	+3.0
25%	49.7	+6.2
10%	51.5	+8.0
0%	53.2	+9.7

Table 13 sets Proposition 2's closed form against the exact rule at the same five read costs, once with draws rounded down to whole candidates and once with divisible draws.

Table 13: Proposition 2's prediction against the exact read-cost sweep, twelve cells, pass^4 (%).

Read cost ρ	Observed, exact rule	Predicted, draws rounded down	Gap	Predicted, divisible draws
100%	43.6	40.6	-3.0	47.4
50%	46.5	46.0	-0.5	51.6
25%	49.7	50.0	+0.3	54.3
10%	51.5	52.3	+0.8	56.1
0%	53.2	55.3	+2.1	58.5

Q VERIFIER QUALITY

Table 14 is the full verifier ladder behind Table 2, one row per verifier and one column per rule that uses it. A text model of the transcript raises AUC from 0.63 to 0.69 on retail and from 0.88 to 0.94 on telecom and moves PATIENCE from 42.9 to 43.2 mean pass^4; the label moves AVA+menu to 44.8.

Table 14: Mean pass^4 of the verifier-dependent rules under three verifiers, twelve cells.

Verifier	AUC (retail/airline/telecom)	Verify cascade	AVA	AVA+menu	PATIENCE	Draw first, verify
Base agent, no verifier				41.4		
Transcript statistics	0.63/0.63/0.88	42.5	42.4	43.2	42.9	42.8
Statistics + transcript text	0.69/0.69/0.94	42.5	42.3	43.2	43.2	42.9
Oracle (ceiling)	1.00/1.00/1.00	44.2	44.0	44.8	44.2	43.6

Two things to notice. Adding the transcript text to the verifier raises its AUC in every domain but moves the rules by at most 43.2 against 42.9 for PATIENCE, because the rules pay for the text they read. The label itself, the best any verifier could be, adds less than two points to every rule that reads whole candidates. At this budget, verifier accuracy is worth about a point to any rule that pays for its reads.

R CROSS-DOMAIN TRANSFER

Table 15 reports PATIENCE trained on one domain and applied to another without retraining, against the in-domain controller.

Table 15: Cross-domain transfer of PATIENCE, mean pass^4 on each target domain.

Target domain	In-domain (5 folds)	Trained on retail	Trained on airline	Trained on telecom
retail	59.5	in domain	58.8	58.6
airline	40.6	41.9	in domain	42.0
telecom	28.7	29.3	29.9	in domain

Every transferred controller lands within a point of the in-domain one, and on airline and telecom the transferred controllers are slightly above it. What transfers is the order of operations, sample early and vote late, not anything about the domain. The takeaway is that the controller can be trained once and reused.

S BUDGET SURFACE AND PER-TASK GAINS

Figure 7 shows the same sweep as a surface over budget and k and, task by task, where PATIENCE's gains over the base agent come from.

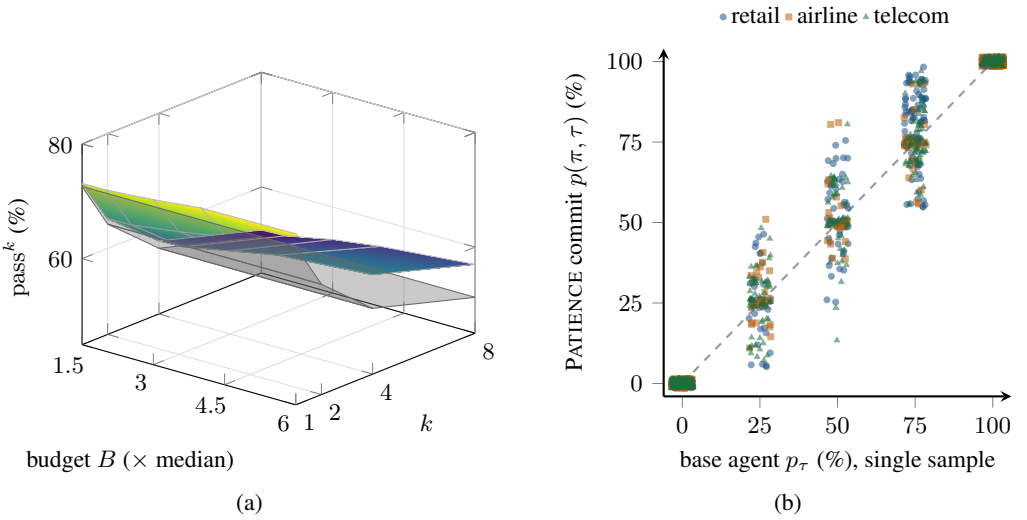

Figure 7: Panel (a), PATIENCE's pass^k over budget and k on retail against the base agent's plane; panel (b), per-task commit probability against the base agent's, all pairs, jittered.

The surface rises with budget at every k and the gap to the base agent's plane is widest at k=8, where the base agent has decayed furthest. The scatter shows where the gains live, among the sometimes-tasks in the interior; tasks at zero or one cannot move. The takeaway is the ceiling argument of Section 1 seen task by task.

T GAINS BY MODEL

The retail cells order cleanly by base strength, which the main text summarizes in one sentence and this section reports in full. On retail the gain falls monotonically with the base agent's strength, from +5.1 points for gpt-4.1-mini (base 46.9%) through +4.1 for o4-mini and +2.1 for gpt-4.1 to +0.9 for claude-3.7-sonnet (base 65.2%), as the ceiling argument predicts, since a weaker agent has more sometimes-tasks to move. It does not carry to airline and telecom, where no model gains more than 1.4 points.

U PER-DOMAIN CONTRASTS

Table 16 repeats the main contrasts within each domain, with intervals from a cluster bootstrap over that domain's tasks.

Table 16: Paired contrasts by domain, pass^4 points, cluster bootstrap over tasks; * excludes zero.

Contrast, pass^4 points	retail (114 tasks)	airline (50)	telecom (114)
PATIENCE minus base	+3.03 [+2.01, +4.09]*	+0.60 [-0.23, +1.57]	+1.01 [+0.31, +1.77]*
menu effect (AVA)	+0.92 [+0.34, +1.57]*	+0.71 [+0.04, +1.49]*	+0.82 [+0.19, +1.50]*
menu effect (PATIENCE)	+1.83 [+0.86, +2.87]*	-0.21 [-0.96, +0.74]	+0.11 [-0.63, +0.87]
target effect	+0.92 [+0.19, +1.70]*	-1.48 [-2.99, -0.31]*	-0.29 [-0.96, +0.39]
rule effect	+1.40 [+0.48, +2.31]*	-1.26 [-2.60, -0.20]*	-0.99 [-1.85, -0.16]*
best-of-k minus base	+2.63 [+1.73, +3.56]*	+1.78 [+0.47, +3.30]*	+1.32 [+0.60, +2.07]*

Retail carries the largest effects and airline, with fifty tasks, the widest intervals. One row deserves a plain statement. The target effect is positive on retail, negative on airline, and null on telecom, so the null the body reports is an average of opposite signs, and tuning for consistency helped one domain and hurt another. Best-of-k beats the base agent in all three domains, which is the operations-beat-samples finding at domain level.

V SPEND BY DOMAIN

Table 17 shows that every rule ran under the same byte meter, by domain. The base agent spends 12.5 KB per run, best-of-k 29.1 KB, AVA+menu 25.3 KB, and PATIENCE 25.7 KB against a mean budget of 35.4 KB; PATIENCE matches best-of-k with about twelve % fewer bytes, the one place learning paid, in cost rather than reliability. The released trajectories also carry Sierra's own dollar accounting for the agent's calls, from which the price of a transcript kilobyte ranges from 1.52 to 29.74 thousandths of a dollar across the four models. PATIENCE's extra 13.2 KB per run therefore costs between $0.020 and $0.39 per conversation for its +1.76 points of pass^4, a price an operator can weigh against the cost of a failed conversation.

Domain	B	Base agent	Best-of-k	Verify cascade	AVA	AVA+menu	PATIENCE
retail	36.7	12.6	30.1	24.5	24.6	25.1	27.6
airline	28.9	10.4	23.2	20.3	19.8	19.9	20.4
telecom	40.6	14.5	34.0	29.6	30.3	30.8	28.9

Table 17: Mean kilobytes spent per run by domain and rule, with the per-domain mean budget B.

No rule exceeds its domain's budget, and PATIENCE spends less than best-of-k in every domain, most on telecom. The takeaway is that every comparison in the body is at equal or lower spend for the rule being credited.