How Does Test-Time Compute Affect Agentic Forecasting?

Anonymous authors
Paper under double-blind review

Abstract

Test-time compute scaling has been explored extensively in easily verifiable domains such as math and code. However, much less is known about how test-time budget should be spent for forecasting future events, where no test-time verifier is available. We study the problem of compute allocation for *agentic forecasting* by comparing three multi-agent policies that spend inference-time compute in different ways: static depth (*Predictor–Critic*), static breadth (*Ensemble + Aggregator*), and adaptive routing (*Hierarchical Orchestrator*). We perform a contamination-controlled backtesting evaluation on 228 label-balanced binary questions from 12 ForecastBench partitions, all resolved strictly after every base model's knowledge cutoff, under a shared web-filtering layer that screens retrieved pages for forecast leakage and post-resolution contamination. In the `gpt-5.4-mini` sweep, adaptive routing occupies the entire cost–accuracy Pareto frontier (80.7% at $\$0.18$/q vs. 78.1%/$\$0.90$ Ensemble and 76.8%/$\$1.67$ Predictor–Critic). The same Pareto ordering replicates on `gpt-5.4-nano`, where the Orchestrator is about $13\times$ cheaper than the top Ensemble with no statistically significant accuracy gap; the Orchestrator's cost–quality advantage further extends to other base models (Claude Sonnet 4.5, DeepSeek v4 Flash, Gemini 2.5 Flash), so the result is not a single-model artifact. A two-stage mechanism diagnostic explains this win as *selective delegation*: on three of five base models, the Orchestrator allocates more compute to questions where its cheap direct baseline is uncertain, and this uncertainty can predict where extra compute helps. Adaptive routing wins not by spending less on every question, but by concentrating compute on the ones where it pays off. We release the benchmark, architecture implementations, pipeline, and analysis code.

1 Introduction

Test-time compute scaling, which spends more inference-time computation on a single query through search, deliberation, self-consistency, or multi-agent debate, has emerged as a powerful complement to model scaling (Snell et al., 2024; Wei et al., 2022; Wang et al., 2023; Du et al., 2023; Lifshitz et al., 2025; Kim et al., 2026). Its most convincing successes, however, come from domains with test-time *verifiability* (OpenAI et al., 2026; Guo et al., 2025). In mathematics, code, and many structured reasoning tasks, candidate solutions can often be checked before an answer is returned: programs can be executed, derivations can be inspected for consistency, and independent reasoning paths can be compared against known constraints. Additional compute therefore supports a search-and-selection process in which incorrect candidates can be generated, detected, and discarded.

Forecasting provides a qualitatively different setting for test-time compute scaling. LLM forecasting systems are increasingly being used to predict unresolved future events and time-indexed quantities from incomplete evidence. This literature is broad and includes a variety of approaches: retrieval-augmented forecasters for open-ended future questions (Halawi et al., 2024; Chandak et al., 2026), ensemble and crowd-aggregation approaches (Schoenegger & Park, 2023; Schoenegger et al., 2024), dynamic forecasting across diverse real-world domains (Karger et al., 2025), tool-using agents for temporal event prediction (Ye et al., 2024), and quantitative benchmarks for calibrated, non-binary forecasts (Qin & Andriushchenko, 2026).

The key difference from mathematics and code is that the target outcome is unobserved at decision time. A forecasting agent may retrieve more evidence, generate competing hypotheses, elicit

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.

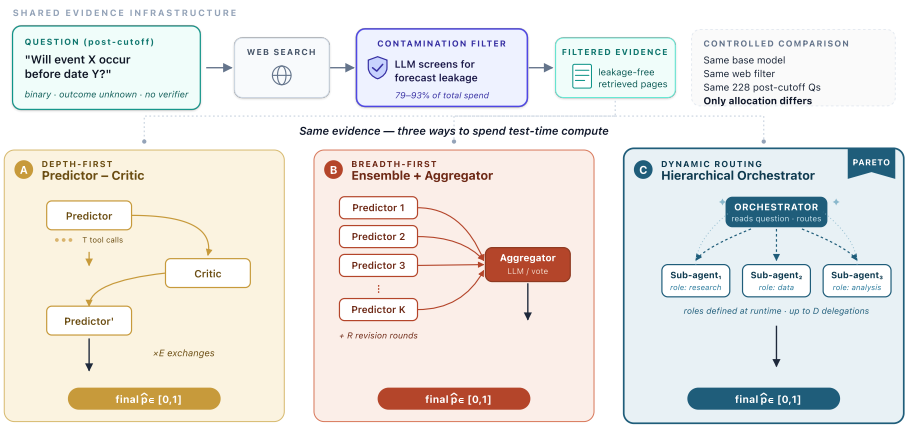


Figure 1: Evaluation pipeline and compute-allocation policies. A shared contamination-controlled retrieval interface feeds three aligned policies: static depth, static breadth, and adaptive routing.

independent forecasts, critique assumptions, or aggregate multiple estimates; however, it cannot observe a correctness signal for any candidate forecast before the event resolves. Additional inference-time compute therefore cannot instantiate the same loop of generation and verification that succeeds in verifier-rich domains. Instead, it can only improve the agent's information state or reduce decision errors conditional on that information, while the realized outcome remains subject to irreducible epistemic uncertainty. This makes it unclear whether test-time compute gains observed on verifiable reasoning benchmarks should transfer to forecasting.

When no test-time verifier is available, what does additional inference compute buy, and how should it be spent across questions, uniformly or selectively on the ones where it pays off?

This question is especially delicate because LLM forecasting is usually retrieval-augmented: agents gather external evidence before predicting (Halawi et al., 2024; Karger et al., 2025). In retrospective backtests, retrieval can leak the resolved outcome through post-resolution pages; even prospectively, agents can outsource the forecast by reading prediction markets or analyst probability quotes rather than reasoning from primary evidence (Paleka et al., 2025; Dai et al., 2025). We therefore evaluate all models under a shared web-filtering stack that targets outcome leakage in retrospective backtests and, at inference time, can exclude market odds, analyst probabilities, or other disallowed sources that would bias the model's own forecast (Section 3) (Paleka et al., 2025).

We study test-time scaling in forecasting through the lens of compute allocation. We instantiate three test-time **compute-allocation policies**: decision rules that distribute inference-time compute across web-based evidence gathering, independent predictors, critique, and delegation (see Appendix E.1). We evaluate all policies under a single contamination-controlled pipeline, which we make publicly available together with the benchmark, model outputs, and analysis code. Our design isolates the effect of compute allocation by holding fixed the base model, web-filtering layer, cutoff date, metrics, and compute-accounting rules across all reported runs. We make the following contributions.

1. **A framework for evaluating compute allocation in verifier-free forecasting.** We curate a balanced post-cutoff benchmark of 228 binary ForecastBench questions and implement a web-evidence filtering layer that screens retrieved pages for forecast leakage and post-resolution contamination in the multi-agent setting; a metadata-based proxy audit estimates that it blocks roughly 90–92% of likely leakage signals. On this benchmark we evaluate three aligned compute-allocation policies: static depth (Predictor–Critic), static breadth (Ensemble + Aggregator), and adaptive routing (Hierarchical Orchestrator), under the same questions, tools, prompts, contamination safeguards, and compute accounting. The complete sweep is run with both `gpt-5.4-nano` and `gpt-5.4-mini`, so shared findings can be separated from model-specific capability or formatting effects.
2. **Adaptive routing on the cost–quality frontier, with retrieval as the dominant cost component.** The Hierarchical Orchestrator occupies the cost–accuracy Pareto frontier on every base model in our sweep, matching or beating both static-depth (Predictor–Critic) and static-breadth (Ensemble) architectures at substantially lower cost. The ordering replicates across two complete model sweeps

(`gpt-5.4-mini`, `gpt-5.4-nano`) and across a five-model Orchestrator scan, ruling out a single-model artifact. Across configurations, the contamination-controlled web filter dominates total spend, identifying retrieval-side compute, not agent deliberation, as the systems bottleneck. The cost–quality ordering carries over to probabilistic outputs once calibration is normalized post-hoc. Full numbers, statistical-significance tests, and per-configuration tables in Section 4.

3. **A mechanistic explanation of the cost–quality dominance.** We propose the *selective-delegation hypothesis*, in which adaptive routing wins because it concentrates compute on questions where extra compute is more likely to help, and test it via a two-stage diagnostic. We find that on three of five base models, the Orchestrator's per-question LLM compute correlates with direct-baseline uncertainty (a computable proxy for marginal compute value), and that direct-baseline uncertainty correlates with Brier improvement across all five models.

2 Compute allocation in verifier-free forecasting

As established in the introduction (Section 1), forecasting is a verifier-free setting: at inference time the binary resolution $Y \in \{0, 1\}$ is unobserved, so candidate forecasts cannot be checked before they are returned. We therefore treat test-time forecasting as a *compute-allocation problem*: a policy π distributes inference-time actions, including web/tool calls, predictor calls, critique, aggregation, and sub-agent delegations, across the trajectory of answering question q under budget C, and emits a forecast at termination. We compare three such policies: static depth (Predictor–Critic), static breadth (Ensemble + Aggregator), and history-dependent adaptive routing (Hierarchical Orchestrator). The full sequential formalism is in Appendix E.1.

We distinguish between compute that changes the available evidence and compute that reasons over fixed evidence. Once a policy has gathered its evidence, additional reasoning cannot raise accuracy beyond what is, in principle, decidable from that evidence: there is an *evidence-limited accuracy ceiling*. Additional retrieval or broader queries can raise this ceiling; further reasoning over the same evidence may improve decisions but cannot raise the ceiling. Appendix E.2 and Appendix E.4 formalize this distinction through a Bayes-risk decomposition.

This framing yields four empirical predictions that we test:

- **(H1) Adaptive routing can improve the cost–quality frontier** (Section 4.1).
- **(H2) Policies should show diminishing returns toward similar evidence-limited ceilings** under shared retrieval and contamination filtering (Section 4.2).
- **(H3) Retrieval-side compute can dominate total cost** when contamination control is implemented as an evidence-filtering layer (Section 4.4).
- **(H4) Selective-delegation hypothesis.** The cost–quality advantage of adaptive routing can be mechanistically explained by *selective spending* – concentrating compute on the questions where extra compute is more likely to help (easy questions terminate early, ambiguous questions spend more), not just spending less on every question (Section 5).

3 Benchmark, metrics, and compute policies

Benchmark: 228 balanced post-cutoff questions. Our benchmark is 228 binary forecasting questions sub-sampled from the ForecastBench pool (Karger et al., 2025) under three joint constraints: (i) every question's `market_info_close_datetime` falls strictly after the knowledge cutoff of every base model used here (`gpt-5.4-mini`, `gpt-5.4-nano`, Claude Sonnet 4.5, DeepSeek v4 Flash, Gemini 2.5 Flash), with resolutions in February 2026–January 2027; (ii) exactly 114 True / 114 False, removing the majority-class shortcut; (iii) topical mix preserved across 12 ForecastBench partitions (Politics/Geopolitics, Technology, Sports, Economics/Finance, and others). Full dataset curation details, including per-partition counts and the topical breakdown, are in Appendix G.

Contamination safeguards. We address the four pitfalls catalogued by Paleka et al. (2025): **(i) Temporal leakage**, by using a *web-filter LLM* to inspect every retrieved page and remove any page containing timestamps or event references after the question-specific cutoff date, which we derive from the ForecastBench market metadata (the midpoint before `market_info_close_datetime`); **(ii) Forecast contamination**, by removing pages containing explicit probability estimates, prediction-market odds, analyst forecasts, or price targets; **(iii) Backtesting artifacts**, by using

`market_info_midpoint_datetime`, rather than the later `close_datetime`, so agents forecast from an information state before the market has closed; and **(iv) Skewed base rates**, by curating a 50/50 True/False set (Section 3).

As an audit of this filter, we run a metadata-based proxy recall check over 847431 page-level filter decisions from the nano and mini runs. Pages whose metadata indicates a likely leakage signal are blocked with proxy recall 89.7% (Wilson 95% CI $[89.5, 89.8]\%$), rising to 92.1% ($[91.9, 92.3]\%$) under a stricter metadata definition. Because the metadata proxy is not semantic validation, we also drew a blinded 200-page sample and labelled each page for outcome leakage and forecast outsourcing. The human audit estimates recall at 0.86 (95% CI $[0.72, 0.97]$), precision at 0.38 ($[0.26, 0.50]$), and residual contamination among the evidence reaching the agents at 6.5% ($[1.6, 13.1]\%$). Thus, the filter catches most pages annotated as contaminated, although some contamination still reaches the agents. Appendix U gives the full metadata-audit breakdown and candidate-miss analysis.

Evaluation metrics and compute accounting. For a benchmark of N questions $\{q_1, \ldots, q_N\}$, let $y_q \in \{0, 1\}$ denote the realized resolution of question q, and let policy π emit a probabilistic forecast $\hat{p}_q \in [0, 1]$ together with a parsed binary answer $\hat{y}_q \in \{0, 1\} \cup \{\perp\}$, where $\perp$ indicates an unparseable output.

Forecast quality. Our headline decision metric is strict accuracy, $\mathrm{Acc}(\pi) = N^{-1} \sum_q \mathbb{1}[\hat{y}_q = y_q]$, with $\hat{y}_q = \perp$ counted as incorrect. We also report Brier score, $\mathrm{Brier}(\pi) = N^{-1} \sum_q (\tilde{p}_q - y_q)^2$, setting $\tilde{p}_q = 0.5$ when $\hat{y}_q = \perp$, so that probabilistic quality is not hidden by thresholding. Conditional versions of both metrics are computed after dropping $\perp$, always paired with the answer rate $\#\{q : \hat{y}_q \neq \perp\}/N$.

Cost and uncertainty. We report seven compute proxies per configuration: total USD cost, agent-only LLM cost, total tokens, prompt tokens, LLM API calls, measured tool calls, and theoretical LLM calls (Section 3.1). We compare paired binary outcomes with an exact McNemar test and report bootstrap (10 000-resample) 95% confidence intervals for accuracy differences across architectures.

3.1 Architectures

We compare three multi-agent compute-allocation policies. All three take the same question, call the same web-search tool through the same contamination filter, share the same base model, and output a single probability $\hat{p} \in [0, 1]$; they differ only in *how* they spend additional inference-time compute. **(A) Predictor–Critic** (depth-first) refines a single predictor's forecast over E critic exchanges with predictor tool budget T, in the spirit of Tree-of-Thoughts (Yao et al., 2023a) and Reflexion (Shinn et al., 2023). **(B) Ensemble + Aggregator** (breadth-first) draws K independent predictors with R optional inter-agent revision rounds and combines them via LLM aggregator or confidence-weighted vote, generalising self-consistency (Wang et al., 2023) and multi-agent debate (Du et al., 2023; Liang et al., 2024). **(C) Hierarchical Orchestrator** (dynamic routing) delegates up to D sub-tasks to sub-agents with tool budget T whose roles it defines at runtime, following LATS-style adaptive search (Zhou et al., 2024) and AutoGen/MetaGPT-style orchestration (Wu et al., 2024a; Hong et al., 2024). The five integer knobs (T, E, K, R, D) are the only architectural degrees of freedom; everything else, including base model, prompts, filter, tool suite, and metric implementation, is shared, and we sweep them to trace each architecture's efficient frontier (Section 4). Full per-architecture descriptions, delegation modes, and design rationale are in Appendix C.

Compute accounting. Worst-case per-question LLM- and tool-call upper bounds for each architecture, derived from the shared ReAct-style loop (Yao et al., 2023b), are deferred to Appendix A and used to compute utilisation $\mathrm{actual}/\mathrm{theoretical}$ in Section 4.3.

3.2 Experimental Setup

The main experiment is a controlled OpenAI `gpt-5.4-mini` sweep over the three compute-allocation policies in Figure 1. Ensemble varies $K \in \{1, 4\}$ and $R \in \{1, 2, 4\}$; Predictor–Critic varies $T \in \{2, 5\}$ and $E \in \{0, 1, 2, 5\}$; and the Orchestrator varies $D \in \{0, 2, 3, 5\}$ and $T \in \{1, 2\}$, using direct tool access at $D{=}0$ and freeform delegation for $D \geq 2$. The main analysis reports 22 `mini` configurations: 6 Ensemble, 8 Predictor–Critic, and 8 Orchestrator. We use OpenAI `gpt-5.4-nano` as a full-sweep replication in the appendix, and report an additional Orchestrator-only cross-model check over complete delegation-enabled configurations from Claude Sonnet 4.5, DeepSeek v4 Flash, Gemini 2.5 Flash, `gpt-5.4-mini`, and `gpt-5.4-nano` in Section 4.5. All runs use the same benchmark, tools, web-filtering layer, metrics, and compute accounting. Further setup details, including baselines, variance estimation, and implementation notes, are in Appendix Q.

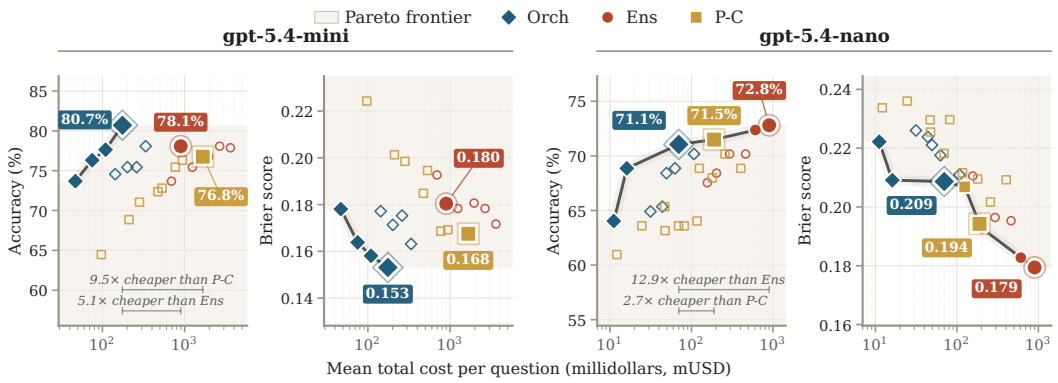


Figure 2: Cost–quality Pareto across both `gpt-5.4` sweeps. The Orchestrator occupies the low-cost accuracy frontier; Brier scores provide a complementary probabilistic comparison.

4 Empirical findings: allocation, saturation, and the filter

We use the contamination-controlled benchmark and compute accounting of Sections 3 and 3.2 to evaluate the three compute-allocation policies of Section 2: static depth (Predictor–Critic), static breadth (Ensemble + Aggregator), and adaptive routing (Hierarchical Orchestrator). The main analysis is the controlled `gpt-5.4-mini` sweep over 22 configurations (6 Ensemble, 8 Predictor–Critic, 8 Orchestrator). All three policies share the same questions, tool stack, contamination filter, and cost accounting; only the allocation rule varies.

Cost convention. Unless otherwise stated, cost refers to *total pipeline cost*: agent/model calls plus the web-filter calls required for contamination-controlled backtesting. We report agent-only cost, token, and call-count proxies separately in the appendix; the qualitative Pareto ordering is preserved under all three (Appendix J). This distinction matters because part of the web-filter cost is specific to backtesting, especially the temporal-cutoff component, while the forecast-outsourcing filter remains relevant for prospective deployment. Retaining that filter still requires inspecting retrieved page content, so removing only the temporal backtesting check would not reduce the cost to the agent-only proxy. Prospective deployment costs would therefore lie between the total and agent-only proxies, and may remain closer to total pipeline cost when forecast-outsourcing filtering is retained.

Section 2 motivates three qualitative hypotheses about what we expect to observe when these policies run head-to-head, and the experiments are designed to test each in turn: (**H1**) adaptive routing can occupy the cost–quality frontier because it can condition compute on the realized history; (**H2**) under a shared retrieval-and-filter stack, all policies are expected to saturate as their budgets grow; and (**H3**) when contamination control is implemented as page-level filtering, retrieval-side filtering may dominate the cost accounting. We also flag a statistical caveat at the outset: with $N{=}228$ and an inter-architecture peak-accuracy spread of at most 4.0 percentage points, the pairwise raw-accuracy differences among the three best configurations are not statistically significant (Appendix R). The robust empirical claim of this section is therefore *cost–quality dominance of adaptive routing*, not raw-accuracy superiority.

The section is organized around four findings followed by a robustness check. Section 4.1 establishes the headline cost–quality result (Figure 2, Table 1); Section 4.2 shows that within each architecture additional compute saturates and is sometimes non-monotone (Figure 3); Section 4.3 identifies the underlying mechanism via a utilisation gap (Table 2); Section 4.4 locates where the systems cost actually goes (Table 3); and Section 4.5 verifies that the headline result is not a single-model artifact . Compute-proxy robustness (token- and call-based Paretos), calibration recovery, an oracle architecture router, descriptive saturation fits, and the full `gpt-5.4-nano` sweep are reported in the appendix and surfaced in the Discussion (Section 6) where they bear on the narrative.

4.1 Adaptive routing dominates the cost–quality frontier

Finding. The Hierarchical Orchestrator occupies the entire observed cost–accuracy Pareto frontier. Its best configuration is also the cheapest of the three best-per-architecture points, by a factor of $5.1\times$ relative to Ensemble and $9.5\times$ relative to Predictor–Critic.

Table 1: Best configuration per architecture on the 228-question benchmark for the `gpt-5.4-mini` and the `gpt-5.4-nano` sweep. Bold entries mark the best value within each model block.

Architecture	Best config	Acc. (%)	Brier	Cost ($/q)	Tokens/q (M)	LLM calls/q
`gpt-5.4-mini` (main sweep)						
Pred.–Critic	T=5, E=5	76.8	0.168	1.665	2.31	53.7
Ensemble	K=1, R=2	78.1	0.180	0.897	1.41	25.9
Orchestrator	D=2, T=2, freeform	**80.7**	**0.153**	**0.175**	**0.22**	**9.0**
`gpt-5.4-nano` (replication)						
Pred.–Critic	T=5, E=2	71.5	0.194	0.189	0.98	24.1
Ensemble	K=4, R=4	**72.8**	**0.179**	0.899	5.88	135.2
Orchestrator	D=3, T=2, freeform	71.1	0.209	**0.070**	**0.32**	**12.9**

The best Orchestrator configuration (D=2, T=2, freeform) reaches 80.7% accuracy at $0.175/question. The best Ensemble configuration (K=1, R=2) reaches 78.1% at $0.897/question, and the best Predictor–Critic configuration (T=5, E=5) reaches 76.8% at $1.665/question. The same pattern also appears in probabilistic forecast quality: among the configurations with the best accuracy for the three architectures, the Orchestrator records the lowest Brier score, at 0.153, compared with 0.180 for Ensemble and 0.168 for Predictor–Critic. Appendix M reports raw and calibrated proper scores for both model sweeps. In the `mini` sweep, the Orchestrator therefore provides the strongest overall cost–quality result: the highest accuracy, the lowest Brier score, and the lowest cost among the best configurations of the three architectures. This conclusion does not depend on web-filter pricing: the same Orchestrator-only frontier appears under agent-only cost, LLM-call count, and token-count proxies (Appendix J). The Orchestrator's best configuration uses $0.021 agent-only cost and 9.0 agent-side LLM calls per question, against the best Ensemble's $0.050 and 25.9 calls. Under total-token accounting (agent plus web filter), the corresponding counts are 219k vs. 1.41M tokens. The comparison also extends beyond the three agentic architectures. Appendix T reports no-tool, direct tool-use, and native reasoning-effort baselines.

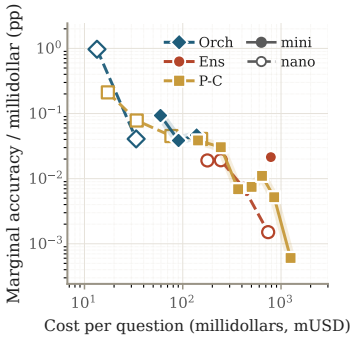


Figure 3: Marginal gain per millidollar along each efficient frontier. Early transitions carry most of the benefit; later compute saturates.

Because the pairwise peak-accuracy differences are not statistically significant (Appendix R), our headline is cost–quality dominance rather than raw-accuracy superiority. Given this architecture-level result, the next question is whether additional compute continues to improve accuracy within each architecture.

4.2 Within-architecture scaling: diminishing and non-monotone returns

Finding. Inside every architecture, accuracy is non-monotone in budget: adding more predictors, exchanges, or delegations beyond a small sweet spot does *not* reliably improve performance. Frontier gains are concentrated in a small number of low-budget transitions, especially for Predictor–Critic (Figure 3). The clearest examples of non-monotonicity are: in Ensemble, going from K=1, R=1 to K=4, R=4 raises accuracy by 4.2 percentage points but multiplies cost by 5.2×; in Predictor–Critic, moving from T=2, E=5 to T=5, E=5 adds substantial cost for only 0.5 percentage points; in Orchestrator, jumping beyond D=2, T=2 leaves accuracy flat or worse at higher cost. The full per-architecture heatmaps are in Appendix K; descriptive saturation fits to each architecture's efficient frontier and their robustness to the fitting procedure are reported in Appendix O (Appendix O.3) and Appendix P.

This is the systems half of the answer to "why doesn't more compute buy more accuracy?". The complementary question is what the Orchestrator does *differently* that lets it sit at the cheapest point on the joint frontier.

4.3 ACTUAL VS. THEORETICAL COMPUTE BUDGETS

Finding. The Orchestrator's advantage can be partly explained by efficient macro-allocation: it reaches the best cost–quality point with a small useful delegation budget. Larger Orchestrator budgets are only partially used and do not necessarily improve accuracy. Static architectures, by contrast, instantiate their predictor, critic, or ensemble schedules much more mechanically across questions. Table 2 summarizes the main utilisation contrast; the full per-configuration overlay is in Appendix B (Figure 5).

Concretely, the best Orchestrator configuration ($D{=}2, T{=}2$) uses a compact agent-side budget: its measured LLM calls are 9.0 per question, matching the theoretical maximum 9. This is far smaller than the best Ensemble (25.9 calls) and Predictor–Critic (53.7 calls). Increasing the Orchestrator delegation budget does not translate into proportional useful compute or higher accuracy. At $D{=}5, T{=}2$, the theoretical maximum is 21 agent LLM calls, but the empirical mean is 17.6; measured tool calls average 7.6 against a theoretical maximum of 10. This makes H1 concrete: the Orchestrator reaches its best result by finding a smaller useful macro-budget, not by spending more compute. However, macro under-utilization alone cannot distinguish uniform budget management, in which the Orchestrator spends less across all questions, from selective allocation, in which it concentrates compute where extra compute is more likely to help. We test the latter mechanism using direct-baseline uncertainty in Section 5.

Table 2: Theoretical vs. measured LLM calls per question on the `mini` sweep (web-filter calls excluded).

Config.	Theory	Actual	Used
Ens. $K{=}1, R{=}2$	24.0	25.9	108%
PC $T{=}5, E{=}5$	66.0	53.7	81%
Orch. $D{=}2, T{=}2$	9.0	9.0	100%
Orch. $D{=}5, T{=}2$	21.0	17.6	84%

This comparison accounts for the agent-side call budget, but not for the total-cost differences in Table 1. The next section examines where the remaining cost arises.

4.4 THE RETRIEVAL-AND-FILTER BOTTLENECK

Finding. The web-filtering LLM, needed to prevent retrieval of post-resolution contamination or forecast-leaking content, dominates total cost across every configuration in the sweep. Agent reasoning is a minority share of total spend.

The web filter accounts for 83–95% of total spend across the `mini` sweep (median 91.8%). External-evidence handling, not agent deliberation, is the systems bottleneck. The Orchestrator's lower share (83–93%) follows directly from its smaller and history-conditioned retrieval volume; Ensemble and Predictor–Critic, which retrieve at fixed multiples of K and T respectively, sit near the upper end of the range (about 90–95%). This is evidence for H3: when contamination control is implemented as filtering on retrieved pages, retrieval-side compute dominates the accounting. The absolute decomposition in Appendix V shows that this pattern is not only a percentage-share effect: static breadth and depth also incur much larger total dollar costs. We discuss implications for future contamination-aware retrieval in Section 6. The full percentage and dollar decompositions are in Appendix V (Figures 19 and 20). The remaining question is whether the cost–quality dominance result holds outside the `gpt-5.4` model family.

Table 3: Representative `mini` cost split.

Config.	Web share	Total $/q
Orch. $D{=}2, T{=}2$	87.9%	0.175
Orch. $D{=}5, T{=}2$	87.7%	0.336
Ens. $K{=}1, R{=}2$	94.4%	0.897
PC $T{=}5, E{=}5$	91.9%	1.665

4.5 CROSS-MODEL ROBUSTNESS

Finding. The cost–quality dominance of adaptive routing is not a single-model artifact. A delegation-enabled Orchestrator comparison over the selected D settings confirms that orchestration carries the result across base-model families, while the best delegation depth is model-dependent. We scan Orchestrator runs from Claude Sonnet 4.5, DeepSeek v4 Flash, Gemini 2.5 Flash, `gpt-5.4-mini`, and `gpt-5.4-nano`. For this cross-model comparison, we focus on the selected delegation-enabled settings.

Two observations follow from this comparison; the numeric table is in Appendix S. First, delegated orchestration is not an OpenAI-only artifact: `gpt-5.4-mini` reaches $184/228$ correct, while Claude

Sonnet 4.5 and DeepSeek v4 Flash each reach $183/228$. Gemini 2.5 Flash and `gpt-5.4-nano` reach 71.5% and 71.1%, respectively. Second, among the delegation-enabled Orchestrator settings we compare across models, the best configuration is not uniform: `mini` and Claude peak at $D{=}2, T{=}2$; DeepSeek peaks at $D{=}3, T{=}1$; Gemini and `nano` peak at $D{=}3, T{=}2$.
The full `gpt-5.4-nano` sweep shows the same cost–quality pattern. Its top Ensemble is slightly higher in raw accuracy than its top Orchestrator (72.8% vs. 71.1%), but the paired difference is not statistically resolved (bootstrap 95% CI $[-4.4, +8.3]$ pp, McNemar exact $p = 0.68$), and the Orchestrator is about $13\times$ cheaper ($0.070/question vs. $0.899/question; Appendix S). On `nano`, the Orchestrator thus reaches similar accuracy at a fraction of the cost.

5 Selective delegation: routing on direct-baseline uncertainty

Section 4.3 shows that the Orchestrator leaves part of its delegation budget unspent; however, spending less is necessary but not sufficient for the cost–quality win. What matters is where the saved compute goes. The Orchestrator must allocate selectively; investing more on questions where extra compute is likely to pay off, and less on the rest (H4).

Setup. Consider the simplest version of the routing decision, with per-question loss given by the Brier score $\ell(\hat{p}, y) = (\hat{p} - y)^2$. After the initial history h, the policy either stops, emitting the cheap forecast $\hat{p}_0$ and incurring the conditional expected loss $L_0(h) = \mathbb{E}[\ell(\hat{p}_0, Y) \mid h]$, or continues, delegating one or more sub-tasks and then stopping, emitting $\hat{p}_1$ and incurring $L_1(h) = \mathbb{E}[\ell(\hat{p}_1, Y) \mid h]$. Define the conditional *marginal value of compute* $M(h) = L_0(h) - L_1(h)$, so that $M(h) > 0$ exactly when the extra step is expected to reduce loss. Selective routing in the sense of H4 means concentrating delegations on histories where $M(h)$ is large. The Orchestrator, however, has *no access to the loss* at decision time: the resolution Y is unobserved at inference, so the router cannot condition on any correctness signal and in particular cannot route on $M(h)$ directly. At best, it can route on a *proxy*: a signal computable from h at decision time that predicts $M(h)$.

Proxy: direct-baseline uncertainty. We use the uncertainty of the cheap forecast itself, $u_0 = 1 - |2\hat{p}_0 - 1|$, which is 1 at $\hat{p}_0 = 0.5$ and 0 when the cheap forecast is certain. Two properties make it a natural proxy. First, it tracks the room for improvement: since the Brier loss is nonnegative, $M(h) \le L_0(h)$, and a calibrated cheap forecast has expected loss $\hat{p}_0(1 - \hat{p}_0)$, which increases with u_0. Second, it is available to the router in principle.

Two-stage decomposition. With u_0 as the proxy, we test H4 through two separate claims.
Stage 1: the Orchestrator's per-question spend correlates with u_0 (does the router behave adaptively?).
Stage 2: u_0 correlates with the ex post marginal value of compute (is it a valid proxy for $M(h)$?).
If both hold, the Orchestrator might be routing more compute onto questions with higher expected marginal value, even though it cannot observe $M(h)$ directly. If only Stage 2 holds, the signal is valid but the router does not use it, identifying headroom for a router that uses the signal as input.

Measurement. We pair $D{=}0, T{=}2$ (the direct baseline) against the headline routed config $D{=}2, T{=}2$ (router) on the same 228 questions across the five base models, and write $\tilde{p}_{0,q}$ and $\tilde{p}_{r,q}$ for the parsed probability forecasts of, respectively, the direct baseline and the routed (r) Orchestrator on question q, in the sense of Section 3. We do not extract the Orchestrator's internal draft, so we compute $u_{0,q} = 1 - |2\tilde{p}_{0,q} - 1|$ from the standalone $D{=}0$ run instead. Stage 1 therefore tests whether routed spend tracks baseline uncertainty. Stage 2 is instantiated as the realized, single question analogue of M under this loss: the paired Brier improvement $\widehat{M}_q^{\mathrm{Brier}} = (\tilde{p}_{0,q} - y_q)^2 - (\tilde{p}_{r,q} - y_q)^2$, where $y_q \in \{0, 1\}$ is the realized resolution, so that $\widehat{M}_q^{\mathrm{Brier}} > 0$ when the routed forecast lands closer to the outcome than the direct one. It is a single realization rather than an estimate of the conditional expectation $M(h_q)$; we use only its rank across questions. The spend variable is the Orchestrator LLM cost, excluding web-filter cost. Both stages use two-sided tests; since H4 predicts a positive spend–uncertainty association for Stage 1, the two-sided p-values there are conservative. For a concrete illustration of both stages, Appendix F.2 reports two audited delegation examples.

Stage 1 result. Table 6 reports the Spearman correlation of u_0 with the routed Orchestrator's per-question LLM cost. Three of five models route on uncertainty at conventional significance: `gpt-5.4-mini` ($\rho = +0.145$, $p = 0.028$), `gpt-5.4-nano` ($+0.144$, $p = 0.030$), and `claude-sonnet-4.5` ($+0.135$, $p = 0.042$). The remaining two do not. Figure 9 shows the binned cost–u_0 relationship on the three positive models.

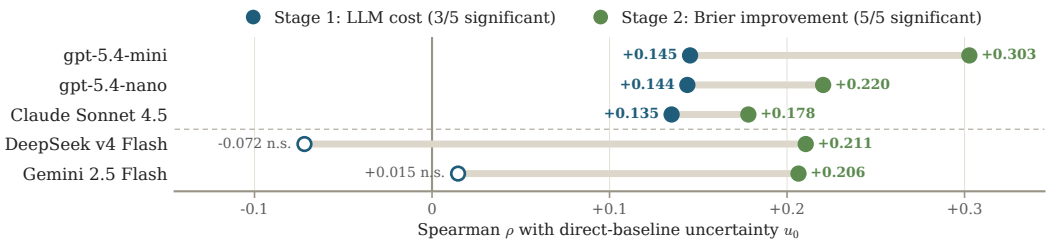


Figure 4: **Selective delegation** (D=2, T=2)**.** Spearman correlations relate direct-baseline uncertainty to the routed Orchestrator's per-question LLM cost (Stage 1) and paired Brier improvement (Stage 2). Filled markers indicate two-sided $p < 0.05$; hollow markers are not statistically significant.

Stage 2 result. u_0 is positively correlated with $\widehat{M}^{\text{Brier}}$ on *every* model, with $\rho \in [0.18, 0.30]$ at $p \leq 0.007$ (Table 6, right): on questions where the cheap baseline is more uncertain, the routed forecast tends to land closer to the realized outcome than the direct one. What varies across models is therefore whether *spend* is also correlated with u_0 (Stage 1), not whether u_0 predicts paired improvement (Stage 2). A comparison across uncertainty quintiles provides a more intuitive view of the Stage 2 result: for `gpt-5.4-mini`, routing turns a wrong direct forecast into a correct one for 15 of 45 questions (33%) in the highest uncertainty quintile, compared with 2 of 47 (4%) in the lowest uncertainty quintile. The five-model binned visualization of this relationship is reported in Figure 6 in Appendix F.1.

Figure 4 summarizes these two stages. Taken together, they show that uncertainty signals where delegation helps, but agents only partly route on it. Brier gains concentrate on the questions the cheap forecast was unsure about yet only gpt-5.4-mini, gpt-5.4-nano, and Claude Sonnet 4.5 spend more when unsure; DeepSeek v4 Flash and Gemini 2.5 Flash do not show a statistically significant relationship. Uncertainty therefore appears to be a useful routing signal, but some agents may leave it unused. The full setup, robustness checks, and limitations are reported in Appendix F.1.

6 Discussion and Conclusion

We presented a controlled study of test-time compute for LLM forecasting on 228 balanced post-cutoff questions. In this verifier-free domain, the binding question is not simply how much inference compute to spend, but where to spend it. Across the `mini` sweep, adaptive routing is the best observed cost–quality allocation: the top Orchestrator reaches 80.7% accuracy at \$0.175/question, while the best static breadth and depth systems cost 5.1× and 9.5× more. The `nano` replication shows the same pattern even when its best accuracy point is an Ensemble, and the cross-model runs suggest that delegated routing is not specific to OpenAI. The mechanism is selective allocation. Static Ensemble and Predictor–Critic schedules spend their budgets on every question, whereas the Orchestrator can stop early or delegate only when additional evidence appears useful. This explains both the low-cost frontier and the diminishing returns within each architecture. Once the filtered evidence has been incorporated, extra critic exchanges or additional parallel predictors often restate the same information rather than raising the evidence-limited ceiling. The Brier results reinforce this story: the best Orchestrator configuration also has the lowest raw Brier among the three configurations.

A natural next step is to turn direct-baseline uncertainty into an explicit allocation policy that learns when to answer directly, retrieve more evidence, delegate, or stop. Randomizing budgets would allow the selective-delegation mechanism to be tested causally. In our backtesting setup, evidence filtering accounts for 83–95% of total spend in the `mini` sweep, so future efficiency gains could come from making contamination-aware retrieval less costly.

These findings are specific to verifier-free forecasting and our contamination-controlled backtesting setup; they should not be interpreted as a general ranking of agent architectures. The architectures make complementary errors (Appendix L), and performance varies across question difficulty and topic (Appendix N). The pre-cutoff experiment provides an additional sensitivity check (Appendix H). Within this scope, the central result is clear: adaptive routing achieves a better observed cost–quality trade-off through selective delegation across questions, while simply increasing compute budgets yields rapidly diminishing returns.

AI USE STATEMENT

Generative AI tools are central to the experimental systems evaluated in this work. LLM agents produce forecasts, retrieve and synthesize evidence, critique predictions, aggregate forecasts, delegate subtasks, and implement the LLM-based web filter, as described in Sections 3, 3.1 and 3.2 and Appendix Q. Generative AI tools were also used to assist with analysis and figure-generation code, debugging, and editing and formatting the manuscript. They were not used to create the benchmark questions or determine their ground-truth outcomes. The authors reviewed all AI-assisted code, figures, and text and verified the reported claims and numerical results against the experiment outputs and analysis scripts. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The paper specifies the benchmark construction, contamination filter, metrics, architecture knobs, model settings, completeness rules, statistical tests, and analysis pipeline in Sections 3 and 3.2 and Appendices G and Q. An anonymized supplementary archive provides the benchmark, architecture implementations, filtering code, and analysis scripts needed to rerun or audit the main experiments, subject to access to the hosted model APIs.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396. Association for Computational Linguistics, December 2023. doi: 10.18653/v1/2023.emnlp-main.761. URL https://aclanthology.org/2023.emnlp-main.761/.

Bradley Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V Le, Christopher Re, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2025. URL https://openreview.net/forum?id=0xUEBQV54B.

Nikhil Chandak, Shashwat Goel, Ameya Prabhu, Moritz Hardt, and Jonas Geiping. Scaling open-ended reasoning to predict the future, 2026. URL https://openreview.net/forum?id=fkYrply0w3.

Hui Dai, Ryan Teehan, and Mengye Ren. Are LLMs prescient? a continuous evaluation using daily news as the oracle, 2025. URL https://arxiv.org/abs/2411.08324.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, Pasquale Minervini, Yanda Chen, Joe Benton, and Ethan Perez. Inverse scaling in test-time compute, 2025. URL https://arxiv.org/abs/2507.14417.

Alex Graves. Adaptive computation time for recurrent neural networks, 2017. URL https://arxiv.org/abs/1603.08983.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming

Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Danny Halawi, Fred Zhang, Chen Yueh-Han, and Jacob Steinhardt. Approaching human-level forecasting with language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24. Curran Associates Inc., 2024. ISBN 9798331314385.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. 2024. URL https://arxiv.org/abs/2310.01798.

Ezra Karger, Houtan Bastani, Chen Yueh-Han, Zachary Jacobs, Danny Halawi, Fred Zhang, and Philip Tetlock. ForecastBench: A dynamic benchmark of AI forecasting capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=lfPkGWXLLf.

Joongwon Kim, Wannan Yang, Kelvin Niu, Hongming Zhang, Yun Zhu, Eryk Helenowski, Ruan Silva, Zhengxing Chen, Srinivasan Iyer, Manzil Zaheer, Daniel Fried, Hannaneh Hajishirzi, Sanjeev Arora, Gabriel Synnaeve, Ruslan Salakhutdinov, and Anirudh Goyal. Scaling test-time compute for agentic coding, 2026. URL https://arxiv.org/abs/2604.16529.

Ali El Lahib, Ying-Jieh Xia, Zehan Li, Yuxuan Wang, and Xinyu Pi. Temporal leakage in search-engine date-filtered web retrieval: A retrospective forecasting case study, 2026. URL https://arxiv.org/abs/2602.00758.

Sang-Woo Lee, Sohee Yang, Donghyun Kwak, and Noah Y. Siegel. Advancing event forecasting through massive training of large language models: Challenges, solutions, and broader impacts, 2025. URL https://arxiv.org/abs/2507.19477.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need, 2024. URL https://arxiv.org/abs/2402.05120.

Zehan Li, Yuxuan Wang, Ali El Lahib, Ying-Jieh Xia, and Xinyu Pi. Simulated ignorance fails: A systematic study of llm behaviors on forecasting problems before model knowledge cutoff, 2026. URL https://arxiv.org/abs/2601.13717.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. pp. 17889–17904, November 2024. doi: 10.18653/v1/2024.emnlp-main.992. URL https://aclanthology.org/2024.emnlp-main.992/.

Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time compute with multiple verifiers. *arXiv preprint arXiv:2502.20379*, 2025.

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E. Gonzalez. Sleep-time compute: Beyond inference scaling at test-time, 2025. URL https://arxiv.org/abs/2504.13171.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: LLMs can predict if they can do better, even mid-generation, 2025. URL https://openreview.net/forum?id=7tOc6h8bea.

Allan H Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology and Climatology*, 12(4):595–600, 1973.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe,

Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2026. URL https://arxiv.org/abs/2412.16720.

Daniel Paleka, Shashwat Goel, Jonas Geiping, and Florian Tramèr. Pitfalls in evaluating language model forecasters. 2025. URL https://arxiv.org/abs/2506.00723.

Jeremy Qin and Maksym Andriushchenko. Quantsightbench: Evaluating llm quantitative forecasting with prediction intervals, 2026. URL https://arxiv.org/abs/2604.15859.

Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (laws)? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=QqVZ28qems.

Philipp Schoenegger and Peter S Park. Large language model prediction capabilities: Evidence from a real-world forecasting tournament. *arXiv preprint arXiv:2310.13014*, 2023.

Philipp Schoenegger, Indre Tuminauskaite, Peter S. Park, and Philip E. Tetlock. Wisdom of the silicon crowd: Llm ensemble prediction capabilities rival human crowd accuracy. 2024. URL https://arxiv.org/abs/2402.19379.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=uLYc4L3C81A.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23. Curran Associates Inc., 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. 2024. URL https://arxiv.org/abs/2402.08115.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024a. URL https://openreview.net/forum?id=BAakY1hNKS.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Scaling inference computation: Compute-optimal inference for problem-solving with language models. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024b. URL https://openreview.net/forum?id=j7DZWSc8qu.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. 2023b. URL https://arxiv.org/abs/2210.03629.

Chenchen Ye, Ziniu Hu, Yihe Deng, Zijie Huang, Mingyu Derek Ma, Yanqiao Zhu, and Wei Wang. Mirai: Evaluating llm agents for event forecasting, 2024. URL https://arxiv.org/abs/2407.01231.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models, 2024. URL https://openreview.net/forum?id=6LNTSrJjBe.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

A Compute-accounting derivation

The main text uses the compact upper bounds in Table 4. This appendix gives the derivation. Every tool-using agent implements the same ReAct-style loop: with tool budget B, it emits at most B tool calls and $B+1$ LLM calls, where the extra LLM call is the final synthesis after the last observation. If the agent emits a final answer early, actual usage is lower; the bounds below are worst-case accounting ceilings.

Let M denote the fixed per-loop tool budget for agents whose budget is not swept. In the reported runs, $M{=}5$ for Ensemble predictors, the Ensemble aggregator, and the Predictor–Critic critic. The swept variable T is used for the Predictor–Critic predictor and the Orchestrator's delegated sub-agents.

Derivation. Each architecture composes ReAct loops in a different pattern. Counting loops, then summing $(B{+}1)$ LLM calls and B tool calls per loop with the appropriate per-loop budget $B \in \{T, M\}$, gives:

- **Ensemble.** K predictors each run the loop in each of $1{+}R$ rounds (initial plus R revisions), then a single aggregator runs one ReAct loop. All predictors and the aggregator use budget M, so

$$\mathrm{LLM}_{\text{theo}} = K(1{+}R)(M{+}1) + (M{+}1), \qquad \mathrm{Tool}_{\text{theo}} = K(1{+}R)M + M.$$

- **Predictor–Critic.** The predictor runs $1{+}E$ ReAct loops (initial plus E revisions), each with swept budget T; the critic runs E ReAct loops, each with fixed budget M. Summing predictor and critic contributions gives

$$\mathrm{LLM}_{\text{theo}} = (1{+}E)(T{+}1) + E(M{+}1), \qquad \mathrm{Tool}_{\text{theo}} = T(1{+}E) + ME.$$

- **Hierarchical Orchestrator.** For $D \geq 1$, the orchestrator itself acts as a router: its available high-level actions are sub-agent delegations, exposed through the same function-calling interface as ordinary tools. It emits at most $D{+}1$ LLM calls: one initial decision, one after each of up to D sub-agent returns, and a final synthesis call. Each delegated sub-agent runs a ReAct loop with swept budget T, contributing up to $T{+}1$ LLM calls and T tool calls per delegation. Thus

$$\mathrm{LLM}_{\text{theo}} = (D{+}1) + D(T{+}1), \qquad \mathrm{Tool}_{\text{theo}} = DT,$$

  understood as the worst case in which all D delegations are used and every sub-agent saturates its tool budget. At $D{=}0$, the orchestrator collapses to a single direct tool-using ReAct loop with budget T, so $\mathrm{LLM}_{\text{theo}}^{D=0} = T{+}1$ and $\mathrm{Tool}_{\text{theo}}^{D=0} = T$.

We use these formulas to compute utilisation rates $\mathrm{actual}/\mathrm{theoretical}$ in Section 4.3. The gap between these worst-case ceilings and observed usage is a mechanism: static schedules instantiate their macro budget on every question, whereas the Orchestrator can stop before consuming all delegations.

Table 4: Theoretical per-question compute upper bounds used for utilisation analysis. These are accounting bounds, not typical-use predictions; measured cost is reported separately in Section 3.

Architecture	Swept knobs	LLM calls upper bound	Tool calls upper bound
Ensemble	K, R	$K(1{+}R)(M{+}1) + (M{+}1)$	$K(1{+}R)M + M$
Predictor–Critic	T, E	$(1{+}E)(T{+}1) + E(M{+}1)$	$T(1{+}E) + ME$
Orchestrator, $D{=}0$	T	$T{+}1$	T
Orchestrator, $D \geq 1$	D, T	$(D{+}1) + D(T{+}1)$	DT

B Full theoretical-vs-actual utilisation overlay

Table 2 in the main text reports representative configurations. Figure 5 shows the full per-configuration overlay for the `gpt-5.4-mini` sweep.

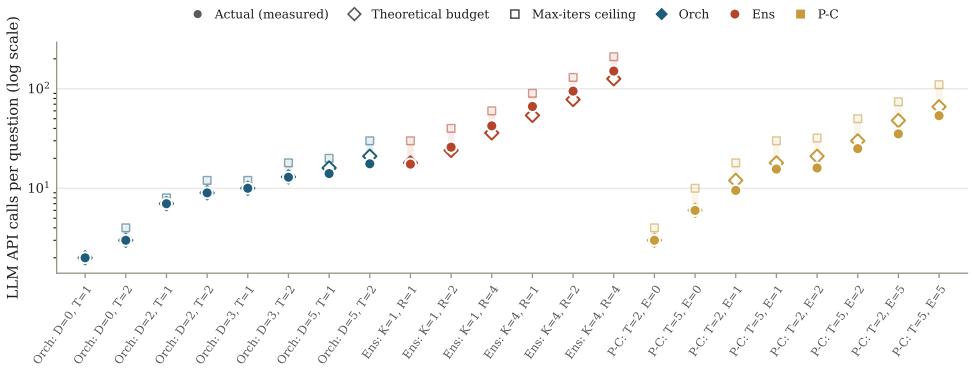


Figure 5: Theoretical vs. actual LLM calls per question on the `gpt-5.4-mini` sweep. Filled discs are measured calls, open diamonds are theoretical per-config budgets, and faded squares are loop max-iters ceilings.

C Architectures: full descriptions

This appendix gives the full per-architecture descriptions condensed in Section 3.1. The three multi-agent compute-allocation policies all take the same question, call the same web-search tool through the same contamination filter, share the same base model, and output a single probability $\hat{p} \in [0, 1]$. They differ only in *how* they spend additional inference-time compute.

A. Predictor–Critic (depth-first). A single predictor with T tool calls produces an initial forecast; a critic challenges its reasoning and the predictor revises over E exchanges, yielding $1 + E$ rounds total. Structurally this instantiates the depth-first deliberate-search style of Tree-of-Thoughts (Yao et al., 2023a) and Reflexion-style self-critique (Shinn et al., 2023), restricted to a fixed-width linear search to keep the budget interpretable.

B. Ensemble + Aggregator (breadth-first). K independent predictors each generate an initial forecast (round 0), optionally exchange information for R further revision rounds (rounds 1 through R, for $1{+}R$ cycles total per predictor), then are combined by either an LLM aggregator or a confidence-weighted majority vote. This generalises self-consistency (Wang et al., 2023) with an explicit inter-agent communication phase along the lines of multi-agent debate (Du et al., 2023; Liang et al., 2024).

C. Hierarchical Orchestrator (dynamic routing). A central orchestrator analyzes each question and dynamically delegates up to D sub-tasks to sub-agents whose roles it defines at runtime based on the question's content and on the output of each previous sub-agent. Each sub-agent receives a tool budget T and has access to the full tool suite (web search, financial APIs, reasoning); the orchestrator is forced to produce a final answer once D delegations have been used. We use the

freeform delegation mode throughout: at $D{=}0$ the orchestrator predicts directly with tool access (no sub-agents); for $D \geq 2$ the orchestrator is free to define each sub-agent's objective from scratch. A separate *predefined* mode (fixed Research / Data / Analysis sub-agents) was implemented but is not part of the reported sweep (Appendix W). The adaptive-routing structure follows the LATS family of agentic search architectures (Zhou et al., 2024) and the orchestration abstraction of AutoGen / MetaGPT (Wu et al., 2024a; Hong et al., 2024), specialised here to forecasting and constrained to a hard delegation budget so that the policy class is comparable to (A) and (B).

Compute knobs. The five integer knobs (T, E, K, R, D) are the only architectural degrees of freedom; everything else, including base model, prompts, filter, tool suite, and metric implementation, is shared. We sweep them to trace each architecture's efficient frontier (Section 4). At $D{=}0$ the Orchestrator predicts directly without delegating. The implementation choice of *freeform* vs. *predefined* sub-agent roles, used only in side experiments, is documented in Appendix W.

D Related Work

We organize prior work into four blocks that align with our three contributions: *(i) LLM forecasting systems and benchmarks* and *(iv) temporal contamination and evaluation validity* position our contamination-controlled evaluation protocol (C1); *(iii) multi-agent and deliberative inference* positions our aligned three-architecture comparison (C2); and *(ii) test-time compute and inference-time scaling* positions our empirical Pareto analysis and descriptive saturation fits (C3).

LLM forecasting systems and benchmarks. Halawi et al. (2024) establish that retrieval-augmented LLMs can approach human forecasters with a fixed inference pipeline (retrieve $\rightarrow$ summarize $\rightarrow$ predict $\rightarrow$ aggregate); we treat this as the natural starting point for our study. Schoenegger & Park (2023); Schoenegger et al. (2024) study LLM ensembles in tournaments, and Karger et al. (2025) provide the dynamic benchmark from which our 228-question dataset is curated. Chandak et al. (2026) take a training-time approach: they synthesise forecasting questions from daily news and fine-tune Qwen3 on the resulting *OpenForesight* dataset using reinforcement learning, showing that their 8B *OpenForecaster* matches much larger proprietary models on held-out future events; Lee et al. (2025) argue similarly that massive forecasting-specific training is now within reach and outline the noisiness, knowledge-cutoff, and reward-design challenges that scaling such training would have to overcome. This line of work is complementary to ours: it scales *training* compute with a single-pass inference step, while we hold the model weights fixed and study how different ways of allocating *test-time* compute affect the same base model. Qin & Andriushchenko (2026) concurrently introduce *QuantSightBench*, evaluating LLM *quantitative* forecasting via prediction-interval coverage on continuous quantities, and report systematic overconfidence in every frontier model they test (`gemini-3.1-pro`, `grok-4`, `gpt-5.4`, all ≥ 10 percentage points short of the 90% coverage target). Relative to this line of work, our paper asks the next question, *once the forecasting pipeline is in place, what does extra test-time compute buy us?*, and we study three structurally distinct ways of spending that compute under a unified budget framework.

Test-time compute and inference-time scaling. Snell et al. (2024) show test-time compute can be more effective than parameter scaling on math; OpenAI et al. (2026) and Guo et al. (2025) demonstrate that large-scale reinforcement learning with verifiable rewards turns inference-time deliberation into substantial gains on math and code; Wu et al. (2024b); Brown et al. (2025); Li et al. (2024) fit inference-time scaling laws for math and code, and report sub-logarithmic returns from repeated sampling, with Schaeffer et al. (2025) giving a recent theoretical account of the observed power-law shape. Closest in spirit to our work, Kim et al. (2026) scale test-time compute for *agentic* coding by representing prior rollouts as compact summaries and selecting/refining among them; their setting still sits in the verifier-rich coding regime (SWE-Bench, Terminal-Bench), whereas our forecasting domain has no test-time verifier at all. Chain-of-thought (Wei et al., 2022) and self-consistency (Wang et al., 2023) are the canonical breadth-style references, and Lifshitz et al. (2025) demonstrates verifier-based search. Adaptive-computation policies include learned halting (Graves, 2017; Schuster et al., 2022), anytime algorithms (Zilberstein, 1996), adaptive self-consistency (Aggarwal et al., 2023), and self-aware mid-generation routing (Manvi et al., 2025). Aggarwal et al. (2023) compare Adaptive-Consistency against uniform self-consistency at equal budgets and report the cost–quality trade-off. Their question is when to stop sampling; ours is how to organize retrieval, critique, aggregation, and

delegation, and which questions deserve more of it. We therefore do not claim adaptive allocation or cost–quality evaluation as novel. Our contribution is the aligned comparison of these agentic policies, the separate agent and retrieval cost accounting, and the test of selective delegation. Lin et al. (2025) broaden the picture by showing that some useful compute can be moved out of test time entirely; this is a useful contrast point because forecasting limits the applicability of sleep-time compute: future-question content is by definition not available in advance. Crucially, Gema et al. (2025) document *inverse* test-time scaling, where additional reasoning compute *degrades* performance on certain tasks; this is directly relevant to our diminishing-returns finding (Section 4.2). Our positioning relative to this line of work is empirical rather than theoretical: we report the first *controlled three-architecture Pareto study* of test-time compute in a verifier-free domain (Section 4), with descriptive saturating-compute fits to each architecture's efficient frontier (Appendix E.5). The architecture ranking is *motivated*, not predicted, by a policy-containment argument (Proposition 1), since whether any specific implemented orchestrator realises the containment advantage on a particular benchmark is itself empirical. The dominant spend in our cost analysis is the contamination-filter LLM, not the agent, a systems observation we discuss in (iv) below.

Multi-agent and deliberative inference. Multi-agent debate (Du et al., 2023; Liang et al., 2024) and self-refinement (Shinn et al., 2023; Madaan et al., 2023) explore sequential argumentation and revision. Tree-of-Thoughts (Yao et al., 2023a) formalises depth-first deliberate search over intermediate reasoning states, structurally close to our Predictor–Critic architecture. ReAct (Yao et al., 2023b) interleaves reasoning with tool calls and is the basis of our agent action loop; LATS (Zhou et al., 2024) unifies ReAct-style tool use with ToT-style search and adds external feedback, structurally close to our Hierarchical Orchestrator. AutoGen (Wu et al., 2024a) and MetaGPT (Hong et al., 2024) are general orchestration frameworks; we specialise their adaptive-policy idea to forecasting and tie it to a formal containment argument. Self-correction without an external signal has been shown to be unreliable (Huang et al., 2024; Stechly et al., 2024), which motivates our use of an external critic and a delegating orchestrator rather than purely introspective refinement. We instantiate these strands as three *aligned* architectures so that depth, breadth, and dynamic routing can be compared under a single budget framework.

Temporal contamination and evaluation validity. Paleka et al. (2025) catalogue evaluation pitfalls, including temporal leakage, retrieval contamination by prediction-market prices, skewed base rates, which we explicitly mitigate. Li et al. (2026) show that simply prompting an LLM to "ignore" post-cutoff information is unreliable, motivating our requirement that all evaluation questions resolve strictly *after* the model's knowledge cutoff (Section 3). Lahib et al. (2026) demonstrate that even date-filtered web search leaks post-resolution contamination evidence, substantially inflating apparent forecasting accuracy; this is the direct empirical motivation for the dedicated contamination-filter LLM in our pipeline (Section 3), and the implementation choice that makes filter-side compute the dominant share of total spend in our cost analysis. Dai et al. (2025) provide a complementary view from the model side, showing that LLM forecasting accuracy degrades systematically as pre-training data ages and that retrieval-augmented prompting only partially recovers the gap, consistent with our observation that, once contamination is controlled, additional reasoning compute on a fixed (filtered) evidence set saturates quickly. Together, these recent results support our position that contamination control is a first-class component of any test-time-compute claim about forecasting.

E A COMPUTE-ALLOCATION FRAMEWORK FOR TEST-TIME FORECASTING

The empirical results in Section 4 show three patterns that, taken together, are not adequately explained by existing test-time-compute scaling theory (Snell et al., 2024; Wu et al., 2024b; Brown et al., 2025): (i) a dynamic-routing policy Pareto-dominates static breadth and static depth on every compute proxy; (ii) all three architectures saturate near a similar accuracy ceiling; (iii) marginal accuracy per extra unit of compute drops by an order of magnitude after one or two points on each architecture's efficient frontier. We give a small decision-theoretic framework that provides qualitative intuition for these three observations. The framework combines standard ingredients, namely policy-class containment and the Bayes risk of decisions under fixed evidence, with a descriptive empirical curve fit on the 228-question Pareto data.

Notation reference. The framework uses the standard random-variable convention: *uppercase* symbols denote random variables, *lowercase* symbols denote their realisations, and a hat ($\hat{\cdot}$) marks a quantity emitted by the policy (an estimate). Empirical metrics in Section 3 are Monte-Carlo estimates of the population quantities below.

Table 5: Notation used throughout Appendix E and the appendices. Empirical analogs (with sample subscripts) are defined in Section 3.

Symbol	Type	Meaning
Q	random variable	A forecasting question
$Y \in \{0,1\}$	random variable	The (unobserved at test time) resolution of Q
q, y_q	realisations	A specific question and its realized resolution
$\hat{y} \in \{0,1\}$	policy output (RV)	Binary forecast emitted by π on Q
$\hat{p} \in [0,1]$	policy output (RV)	Probabilistic forecast emitted by π on Q
$\hat{y}_q$, $\hat{p}_q$	realisations	Policy outputs on a specific question q
$\hat{p}_{\text{cal}}$	post-hoc calibrated	Output of Platt / temperature scaling on $\hat{p}$
π	function	Test-time-compute policy (architecture + config)
$C \geq 0$	scalar	Compute budget (USD per question, or token count)
$\mathcal{A}$	set	Action set: {tool call, sub-agent delegation, predictor invocation, critic exchange}
h_t	sequence	History of evidence + intermediate predictions at step t
$R(C)$	set	Evidence retrieved by π within budget C
$L(\pi, C)$	population scalar	Expected zero–one loss of π at budget C
L_{ev}, L_{rs}, L_{cal}	population scalars	Three components of L (Equation (1))
α_{arch}	population scalar	Saturation ceiling of architecture's scaling-law fit
β_{arch}	population scalar	Compute-recoverable headroom
τ_{arch}	population scalar	Compute time-constant (smaller $\Rightarrow$ faster saturation)

E.1 Compute-allocation policies: formal definitions

Notation convention. We follow the standard random-variable convention: *uppercase* Q, Y denote random variables (a forecasting question and its unobserved-at-test-time resolution); *lowercase* q, y_q denote realisations of those variables on a specific question; and the policy's outputs on that question are $\hat{p}_q \in [0,1]$ (probabilistic forecast) and $\hat{y}_q \in \{0,1\}$ (parsed binary answer). The empirical metrics in Section 3 are Monte-Carlo estimates of the population quantities defined here.

Let Q be a forecasting question and $Y \in \{0,1\}$ its (unobserved at test time) resolution. A *test-time-compute policy* is a budget-constrained sequential decision rule. Let $\mathcal{A}$ denote the action set: {tool call, sub-agent delegation, predictor invocation, critic exchange, aggregation step}. Starting from $h_0 = (Q)$ and budget $b_0 = C$, the policy chooses actions according to a kernel

$$\pi_t(\cdot \mid h_t, b_t) \in \Delta(\mathcal{A} \cup \{\text{TERMINATE}\}),$$

where $h_t = (Q, a_{<t}, o_{<t})$ contains all previous actions and observations, and b_t is the remaining budget. If $a_t \in \mathcal{A}$, the evaluation runner executes the corresponding model, tool, or sub-agent operation, receives observation o_t, pays the measured cost, and updates both h_t and b_t. If $a_t =$ TERMINATE, or if no feasible action remains under the budget, the policy stops at time τ and a terminal map g_π emits

$$(\hat{p}, \hat{y}) = g_\pi(h_\tau), \qquad \hat{p} \in [0,1], \quad \hat{y} \in \{0,1\}.$$

The evidence object $R_\pi(C)$ is the projection of h_τ onto the retrieved pages, filtered tool outputs, intermediate forecasts, and other observations available when the policy stops. Deterministic policies or implementations are included as the degenerate case where each $\pi_t(\cdot \mid h_t, b_t)$ is a point mass.

Every architecture in this paper instantiates such a policy:

- *Predictor–Critic* $\pi_{\text{PC}}^{T,E}$ runs T tool calls, then alternates predictor and critic agents for E exchanges according to a fixed macro schedule.
- *Ensemble* $\pi_{\text{Ens}}^{K,R}$ runs K predictors in parallel and exchanges information for R rounds according to a fixed macro schedule.
- *Hierarchical Orchestrator* $\pi_{\text{Orch}}^{D,T}$ dynamically defines and delegates up to D sub-tasks, with both the *assignment* of each sub-task and the *role* of the sub-agent chosen *conditional* on h_t (the question text and the previous sub-agents' return values). At $D{=}0$ the orchestrator acts directly with tool access.

We call a policy *static at the macro-allocation level* if, conditional on the question Q and step t, its next high-level action distribution does not depend on the realized history h_t, and *adaptive* otherwise:

$$\Pi_{\text{static}} = \big\{\pi : P_\pi(a_t \mid Q, t, h_t) = P_\pi(a_t \mid Q, t)\big\},$$

$$\Pi_{\text{adapt}} = \big\{\pi : P_\pi(a_t \mid Q, t, h_t) \text{ may depend on } h_t\big\}.$$

Thus $\Pi_{\text{static}} \subsetneq \Pi_{\text{adapt}}$: a static macro-allocation policy is a special case of an adaptive policy that ignores h_t, while there exist adaptive policies whose high-level actions genuinely depend on the realized history.

Static at the level of macro compute allocation. In our implementation π_{PC} and π_{Ens} are static at the level of *macro compute allocation*: the number of predictor calls, critic exchanges, and ensemble rounds is fixed in advance by the parameters (T, E) and (K, R), and is not modulated by intermediate outputs. Within each individual ReAct invocation, however, the agent does choose its next tool query in response to prior observations, so PC and Ensemble are not *strictly* static at the fine-grained tool-action level. The relevant distinction for our framework and for the policy-class containment argument in Appendix E.3 is at the macro-allocation level: π_{Orch} is the only policy in our sweep whose macro schedule (which sub-agent is delegated to next, with what role, and whether to delegate at all) depends on h_t.

E.2 Two error decompositions: accuracy and calibration

We organize the analysis around two complementary error decompositions, one for thresholded accuracy and one for the probabilistic forecast.

Accuracy decomposition. Let $L(\pi, C) = \Pr(\hat{y} \neq Y)$ be the population zero–one error of policy π at compute budget C. Each policy induces an evidence set $R_\pi(C)$, the pages, queries, and tool outputs π retrieves under budget C, and the loss splits cleanly into two non-negative pieces:

$$L(\pi, C) \;=\; \underbrace{L^*(R_\pi(C))}_{\text{Bayes risk under } \pi\text{'s evidence}} \;+\; \underbrace{\Delta(\pi, R_\pi(C))}_{\text{decision sub-optimality}}, \tag{1}$$

where $L^*(R_\pi(C))$ is the Bayes risk given the evidence distribution induced by π (Appendix E.4), and $\Delta(\pi, R_\pi(C)) \geq 0$ is the additional error from π failing to make the Bayes-optimal call *given that specific evidence*. Increasing reasoning compute on a fixed evidence set can only reduce Δ, never L^*. Note that $R_\pi(C)$ is in general policy-dependent: different architectures may issue different queries at the same budget, so L^* is itself architecture-conditional rather than a single shared quantity. We argue in Appendix E.4 that under our shared retrieval-and-filter stack the gap between the policies' $R_\pi(C)$ distributions is small, which is why their fitted ceilings end up close (Appendix O.3).

Calibration as a separate axis. Calibration error is *not* an additive component of the zero–one loss in (1): a perfectly-calibrated forecaster can be at chance accuracy and a miscalibrated forecaster can have identical thresholded decisions to a well-calibrated one. Calibration naturally decomposes proper scoring rules. We use the standard Murphy decomposition (Murphy, 1973) of the Brier score,

$$\text{Brier}(\pi) \;=\; \underbrace{\text{Reliability}(\pi)}_{\text{calibration error}} \;-\; \underbrace{\text{Resolution}(\pi)}_{\text{discriminative power}} \;+\; \underbrace{\text{Uncertainty}(Y)}_{\text{base-rate variance}}, \tag{2}$$

in which the Reliability term is the only component the calibration recovery procedure (Appendix M) targets: it is the squared gap between the predicted probability and the observed frequency in each bin, weighted by bin size, and is approximated by the empirical expected calibration error (ECE) we report. ECE is therefore an estimator of the Reliability term alone, not of the full Brier score. Resolution measures how strongly the forecaster separates events with different empirical frequencies, and Uncertainty depends only on the base rate of Y (it is the same for every forecaster on the same benchmark).

The two decompositions are independent and we analyze them separately throughout the paper: accuracy via (1) and the Pareto frontier in Section 4, calibration via (2) and ECE in Appendix M. Combining them into a single composite objective would require a custom loss; we do not.

E.3 ALLOCATION DOMINANCE

Proposition 1 (Adaptive policies weakly dominate static ones)**.** *For any compute budget $C \geq 0$, $\min_{\pi \in \Pi_{adapt}} L(\pi, C) \leq \min_{\pi \in \Pi_{static}} L(\pi, C)$, with strict inequality whenever there exists a positive-probability event in which the optimal action depends on h_t.*

Proof sketch. Containment: any static π_s is also in Π_{adapt}, so the adaptive optimum is at least as good. For strictness, observe that an adaptive policy can re-allocate compute when h_t reveals a sub-task is unproductive (e.g. skip a Data sub-agent when the question is purely qualitative); the static policy must spend the same compute regardless. A standard policy-improvement argument (Graves, 2017; Schuster et al., 2022; Aggarwal et al., 2023; Manvi et al., 2025) converts any positive-probability detection event into a strict inequality. □

Proposition 1 is a *necessary* but not sufficient condition for our empirical headline. It establishes that the optimal adaptive policy can in principle match any static schedule on any compute budget; it does *not* guarantee that any specific implemented orchestrator dominates any specific implemented ensemble or predictor–critic on any particular benchmark, since implementation quality of all three architectures will determine how close each is to the corresponding minimum. Whether our concrete Hierarchical Orchestrator realises the containment advantage on the 228-question benchmark is therefore an *empirical* question, which we test in Section 4.

E.4 DECISION-THEORETIC EVIDENCE CEILING

The *shape* of the saturation pattern is bounded by a standard decision-theoretic argument. Let $\eta(R) = \Pr(Y = 1 \mid R)$ denote the posterior probability of $Y = 1$ given evidence set R. For a binary classification loss, the Bayes-optimal decision error given R is

$$L^*(R) \;=\; \mathbb{E}_R\big[\,\min\big(\eta(R),\, 1 - \eta(R)\big)\,\big]\,, \tag{3}$$

and any decision rule π acting on R satisfies $\Pr(\hat{y} \neq Y) \geq L^*(R)$. In our setting the evidence set depends on the budget, $R = R(C)$, so the smallest population error *achievable by any policy operating on the same evidence* is $L_{\text{ev}}(C) \;\geq\; L^*(R(C))$. Two consequences follow:

- **Similar ceilings under shared evidence infrastructure.** Architectures that share the question set, the retrieval backend, and the contamination filter operate on approximately the same evidence distribution and are therefore bounded by approximately the same $L^*(R(C))$. Different policies may issue different queries and accumulate different evidence sets in any given run, so we expect ceilings to be *similar in expectation but not identical*, consistent with the empirical ≤ 0.04 pairwise gap between the best α values per architecture (Table 1).
- **Reasoning compute on fixed evidence is bounded by Bayes risk.** Doubling E in PC or R in Ensemble at fixed $R(C)$ cannot push accuracy past $1 - L^*(R(C))$. Only compute that changes the evidence (more retrieval, broader queries) can lower the floor.

Why verifier-free is qualitatively different. The split itself is generic to classification; the forecasting-specific point is what test-time compute can do to its two terms. In verifier-rich domains, a verifier produces an additional observation V about candidate correctness (for example, program execution or a symbolic validity check), so the effective evidence becomes $R^+_\pi(C) = (R_\pi(C), V)$. Conditioning on a more informative state cannot increase the Bayes-optimal error in expectation, so the verifier raises the evidence-limited ceiling: $L^*(R^+_\pi(C)) \leq L^*(R_\pi(C))$. Verifier feedback can also reduce Δ by making the implemented selection rule closer to Bayes-optimal on the evidence it has. In forecasting, the realized outcome Y is unavailable at test time and there is no analogous correctness observation for a candidate forecast: extra deliberation on fixed evidence can only target Δ, never L^*. This is the formal sense in which verifier-free reasoning is structurally different from verifier-rich reasoning, and it motivates our focus on *allocation*: which actions raise the evidence-limited ceiling vs. which only target Δ, rather than on aggregate inference budget.

Cost-breakdown as a methodological observation. A separate, empirical observation is that in our implementation the contamination filter accounts for 83–95% of total spend in the `mini` sweep (Figure 20). We do not derive this share from theory because it reflects two implementation choices, namely the use of an LLM-based filter (rather than a regex / smaller-model classifier) and the use of a

relatively cheap base model whose own per-call cost is small compared to the per-page filter cost. The figure therefore quantifies *the cost of running a contamination-controlled forecasting evaluation* under our specific stack, and identifies the filter as a clear systems lever for future cost reduction; it is not an information-theoretic prediction.

E.5 A DESCRIPTIVE SATURATION FIT FOR THE EFFICIENT FRONTIER

The Bayes-risk bound in (3) suggests that, on a fixed retrieval-and-filter stack, accuracy as a function of compute should saturate. We fit a three-parameter exponential saturation *descriptively* to each architecture's efficient frontier:

$$\mathrm{Acc}_{\mathrm{arch}}(C) \;=\; \alpha_{\mathrm{arch}} \;-\; \beta_{\mathrm{arch}}\,\exp(-\,C/\tau_{\mathrm{arch}})\,, \tag{4}$$

where α_{arch} is the asymptotic ceiling, β_{arch} the compute-recoverable headroom, and τ_{arch} the architecture-specific compute time constant.

We emphasise that this is a *descriptive* fit, not a scaling "law" in the strict sense: each architecture's efficient frontier contributes only 3–5 non-dominated points, which is too few to estimate a three-parameter family with tight confidence (Appendix O reports bootstrap CIs that are correspondingly wide). We therefore use (4) as a compact summary of the cost–accuracy curves rather than as a predictive scaling model. The framework supplies two qualitative predictions and one explicit non-prediction:

1. *Approximately shared ceilings.* α_{Orch}, α_{Ens}, α_{PC} should be *approximately* equal because the three architectures share the same question set, retrieval backend, and contamination filter, so they operate on similar (in expectation) evidence distributions and therefore similar Bayes risks $L^*(R(C))$ (Appendix E.4). The framework does not predict exact equality, since different policies may issue different queries and accumulate different evidence sets in any given run; we therefore expect the pairwise gap $\max_{i,j} |\alpha_i - \alpha_j|$ to be small but non-zero.
2. *Faster saturation for adaptive routing.* The history-adaptive policy class *can* reach a target accuracy at no greater compute than either static class (Proposition 1). If this advantage is realized by our specific implementation, the orchestrator's τ should be smaller than either static τ. The proposition does not by itself guarantee this for our concrete implementation; that is an empirical question.
3. *Non-prediction: order between the two static policies.* The framework does *not* order τ_{Ens} and τ_{PC}. Their relative speed depends on per-step costs and on which dimension, width K or depth E, offers more diversity per unit cost in a particular regime. We report the empirical ordering in Appendix O.3 as data, not as a confirmed prediction.

Empirical fits on the 228-question data are reported in Appendix O.3 and Appendix O, with confidence intervals from a non-parametric bootstrap.

E.6 WHAT THIS FRAMEWORK EXPLAINS, AND WHAT IT DOES NOT

The framework offers *qualitative* intuitions for: (a) why adaptive routing has a structural opportunity to dominate (§E.3; whether our concrete implementation realises this opportunity is empirical), (b) why all three architectures saturate near approximately the same ceiling under shared evidence infrastructure (§E.4), (c) why diminishing returns are sharp once the evidence ceiling is approached (§E.5), and (d) why calibration error is best analyzed on a different axis than thresholded accuracy (§E.2, (2)). It does *not* predict the absolute value of α (which depends on the retrieval pipeline and the model's prior knowledge), nor the exact ratio of saturation time-constants (which we estimate descriptively, not predictively).

Cross-model hypothesis: delegation budgets and base-model strength interact. The framework extends to a cross-model comparison under one mild assumption: when only the base LLM changes, the per-question evidence ceiling $L_{\mathrm{ev}}(C)$ is unchanged (same web filter, same retrieval), and only the reasoning sub-optimality term $L_{\mathrm{rs}}(\pi, C)$ shifts: weaker models incur a larger L_{rs} at every C. Because the orchestrator's policy class is strictly larger than either static policy's (Proposition 1), it has more freedom to compensate for a larger L_{rs} by re-routing compute. Two qualitative hypotheses follow:

(i) *Saturation regime: delegation substitutes for raw strength.* For depth budgets large enough that the orchestrator can explore most of its policy class, the per-question accuracy gap between the strong and weak base model should *shrink* relative to the no-delegation case, since both models converge towards the same evidence-bounded ceiling.

(ii) *Pre-saturation regime: delegation amplifies raw strength.* For depth budgets too small to fully exploit the policy class, the strong model extracts more value from each marginal delegation than the weak model does, so the gap should *widen* with D before saturation kicks in.

The cross-model robustness experiment (Section 4.5) is consistent with this picture: among complete non-direct Orchestrator runs, the stronger `gpt-5.4-mini` reaches its best point at $D{=}2, T{=}2$, while `gpt-5.4-nano` and DeepSeek v4 Flash prefer the deeper $D{=}3, T{=}2$ setting. We treat this as qualitative evidence only, because the non-OpenAI delegated sweeps are still incomplete.

We test the framework's quantitative predictions in Appendix O.3; the cross-model generality result above is reported in full in Section 4.5.

F Selective-delegation diagnostic: full results and robustness

This appendix collects the figures, sensitivity analyzes, and robustness panels supporting Section 5.

F.1 Main diagnostic details

In order to convert under-spending into a cost–quality win, the Orchestrator must spend it *selectively*: more on questions where extra compute is more likely to help, less on the rest (H4). We test this directly.

Setup. Consider the simplest version of the routing decision. After a cheap initial history H, the policy either stops (incurring conditional expected loss $L_0(H)$) or pays one extra delegation step (incurring $L_1(H)$). Define the conditional *marginal value of compute* $M(H) = L_0(H) - L_1(H)$. Selective routing in the sense of H4 means concentrating delegations on histories where $M(H)$ is large. However, the Orchestrator has *no access to the loss* at decision time: the resolution Y is unobserved at inference, so the router cannot condition on any correctness signal, and in particular cannot route on $M(H)$ directly.

Two-stage decomposition. We decompose the mechanism into two falsifiable claims: **Stage 1.** The Orchestrator's per-question spend correlates with a computable signal that the orchestrator could plausibly use. *Does the router behave adaptively?* **Stage 2.** That signal is, in turn, correlated with the ex post marginal value of compute. *Is the router's target a valid proxy for $M(H)$?* If both hold, then the Orchestrator might be routing more compute onto questions with higher expected marginal value, even though it cannot observe $M(H)$ directly. If only Stage 2 holds, the signal is valid but the router does not use it, identifying headroom for a router that uses the signal as an explicit input.

The signal. A possible signal is the *direct-baseline uncertainty* $u_0(h) = 1 - |2p_0(h) - 1|$, where $p_0(h) = \Pr(Y = 1 \mid h)$ is the policy's posterior. u_0 is high when the cheap forecast is near 0.5 and low when the cheap policy is highly confident; it is computable from h alone, since the orchestrator runs an internal direct-style draft as part of its decision loop. We pair $D{=}0, T{=}2$ against the headline routed config $D{=}2, T{=}2$ (freeform) on the same 228 questions across the five base models, and instantiate Stage 2 as the realized paired Brier improvement $\widehat{M}_q^{\mathrm{Brier}} = (\tilde{p}_{0,q} - y_q)^2 - (\tilde{p}_{r,q} - y_q)^2$, where $\tilde{p}_{0,q}$ is the direct baseline's ($D{=}0, T{=}2$) parsed probability forecast on question q, $\tilde{p}_{r,q}$ is the routed Orchestrator's ($D{=}2, T{=}2$) parsed forecast, and $y_q \in \{0, 1\}$ is the realized resolution. The spend variable is the orchestrator-side LLM cost, excluding web-filter cost.

Stage 1 result. Table 6 reports the Spearman correlation of u_0 with the routed Orchestrator's per-question LLM cost. Three of five models route on uncertainty at conventional significance: gpt-5.4-mini ($\rho = +0.145$, $p = 0.028$), gpt-5.4-nano ($+0.144$, $p = 0.030$), and claude-sonnet-4.5 ($+0.135$, $p = 0.042$). The remaining two models do not. Robustness across $T \in \{1, 2\}$ and $D \in \{2, 3, 5\}$.

Stage 2 result. u_0 is positively correlated with $\widehat{M}^{\mathrm{Brier}}$ on every model, $\rho \in [0.18, 0.30]$ at $p \leq 0.007$ (Table 6, right; scatter in Figure 6): on questions where the cheap baseline is more uncertain, the

Table 6: Two-stage selective-delegation diagnostic at $D{=}2, T{=}2$. Stage 1: orchestrator LLM cost vs. direct-baseline uncertainty (positive on three of five). Stage 2: direct-baseline uncertainty vs. paired Brier improvement (positive on all five at $p \leq 0.007$)

	Stage 1: $\rho_S(u_0, \text{cost})$		Stage 2: $\rho_S(u_0, \widehat{M}^{\text{Brier}})$	
model	ρ	p (two-sided)	ρ	p (two-sided)
gpt-5.4-mini	$+0.145$	0.028	$+0.303$	< 0.001
gpt-5.4-nano	$+0.144$	0.030	$+0.220$	0.001
claude-sonnet-4.5	$+0.135$	0.042	$+0.178$	0.007
deepseek-v4-flash	-0.072	0.328	$+0.211$	0.004
gemini-2.5-flash	$+0.015$	0.826	$+0.207$	0.002

routed forecast tends to land closer to the realized outcome than the direct one. What varies across models is therefore whether *spend* is also correlated with u_0 (Stage 1), not whether u_0 predicts paired improvement (Stage 2).

Uncertainty-quintile comparison. We define a rescue as a question for which the direct forecast is wrong at the 0.5 threshold and the routed forecast is correct. For `gpt-5.4-mini`, routing rescues 15 of 45 questions (33%) in the highest uncertainty quintile, compared with 2 of 47 (4%) in the lowest uncertainty quintile, corresponding to an odds ratio of 11.25 (two-sided Fisher exact $p = 0.00034$). Fifteen of the 28 rescues occur in the highest uncertainty quintile, and 19 occur in the two highest quintiles. This provides an interpretable, threshold-based view of the Stage 2 relationship. The result is specific to `gpt-5.4-mini` and does not constitute a compute-matched causal comparison.

Interpretation and limitations. Both stages are *rank correlations*, not causal claims: we do not intervene on spend, and the Orchestrator does not observe $\widehat{M}^{\text{Brier}}$ at decision time. On the three models where Stage 1 is significant, the joint pattern (spend correlates with u_0, and u_0 correlates with $\widehat{M}^{\text{Brier}}$) is consistent with the selective-delegation mechanism (H4): the Orchestrator's per-question spend is concentrated on questions where extra compute correlates ex post with larger Brier improvement. On the remaining two models, Stage 2 is positive but Stage 1 is null at the headline budget, indicating headroom for a router that uses u_0 as an explicit input.

Brier decomposition (lens for the informativeness check). For a forecaster that emits probabilities partitioned into bins $\{B_p\}$ with empirical bin frequency w_p, mean forecast p, and within-bin outcome rate $\bar{y}_p$ we have that:

$$\text{Brier} \;=\; \underbrace{\sum_p w_p(\bar{y}_p - p)^2}_{\text{Reliability}} \;-\; \underbrace{\sum_p w_p(\bar{y}_p - \bar{y})^2}_{\text{Resolution}} \;+\; \underbrace{\bar{y}(1-\bar{y})}_{\text{Uncertainty}}.$$

Resolution measures how informative the forecaster's confidence is about the actual outcome distribution. The Mann–Whitney test of $u_0 \mid$ wrong stochastically larger than $u_0 \mid$ right is a one-sided resolution test: informativeness can hold whether or not the baseline is well-calibrated, and we do not test calibration here.

Robustness across routed configurations. The body restricts to the headline routed config $D{=}2, T{=}2$. Tables 7 and 8 re-run Stage 1 (cost) and Stage 2 across all routed budgets we ran: $T \in \{1, 2\}$ and $D \in \{2, 3, 5\}$, paired against matching $D{=}0$ baselines. The headline understates the effect for nano ($\rho = +0.232$, $p < 0.001$ at $D{=}3, T{=}2$) and misses a real mechanism for deepseek at the deepest budget ($\rho = +0.325$, $p < 0.05$ at $D{=}5, T{=}2$). The Stage 2 signal is essentially universal: 17 of 20 cells significant, all 20 positive in sign.

Stage 1 (tokens) across routed configurations. Table 9 repeats the Stage 1 robustness sweep on completion-token count instead of cost. Cost and tokens are nearly perfectly rank-correlated within each model, and the qualitative picture is preserved: positive ρ on mini/nano/sonnet at $T{=}2$, deepseek at the extremes, gemini null throughout.

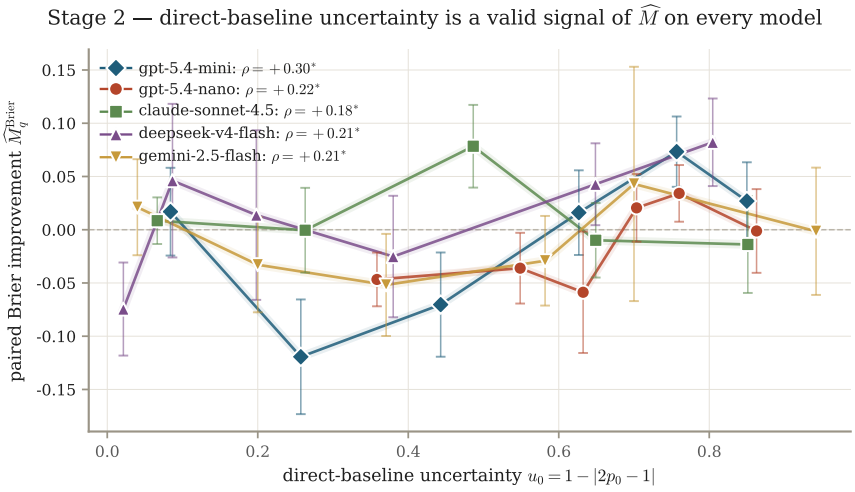


Figure 6: Stage 2 scatter (deferred from Section 5). Paired Brier improvement $\widehat{M}_q^{\text{Brier}}$ as a function of direct-baseline uncertainty u_0, all five base models. Markers are binned means with ± 1 s.e., six bins per model. The relationship is positive on every model at $p \leq 0.007$.

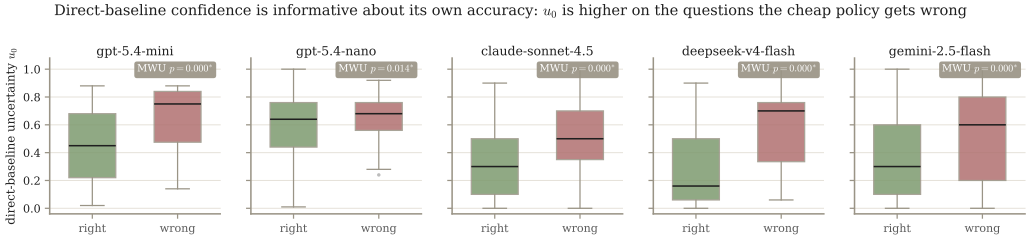


Figure 7: Direct-baseline uncertainty conditional on whether the cheap baseline gets the question right or wrong. On every model, u_0 is stochastically larger when the baseline is wrong (one-sided Mann–Whitney U, $p \leq 0.014$). This is the property that makes u_0 a usable routing signal.

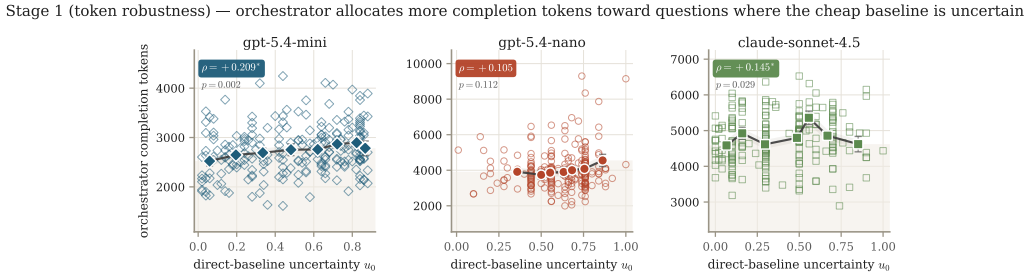


Figure 8: **Stage 1 (token robustness)**. Same three panels as Figure 9-top but with completion-token count on the y-axis. Cost and tokens are nearly perfectly rank-correlated within each model; the conclusions are stable, with completion tokens giving a slightly stronger signal for mini ($\rho = +0.209$, $p = 0.001$).

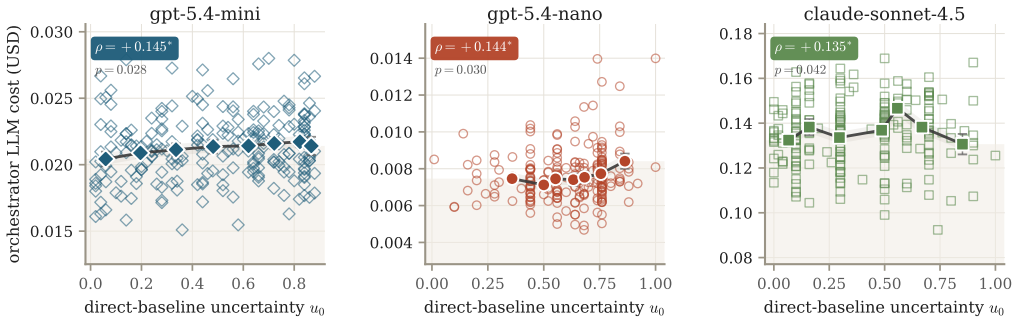


Figure 9: **Stage 1.** Orchestrator LLM cost vs. direct-baseline uncertainty u_0 for the three models with positive Spearman correlation ρ . Markers are binned means with ± 1 s.e.

Table 7: Stage 1 (cost) across routed configurations. Spearman $\rho_S(u_0, \text{cost}_r^{\text{orch}})$ per (D, T). * $p < 0.05$, *** $p < 0.001$.

model	$D{=}2, T{=}1$	$D{=}2, T{=}2$	$D{=}3, T{=}2$	$D{=}5, T{=}2$
gpt-5.4-mini	$+0.075$	$+0.145^*$	$+0.156^*$	$+0.047$
gpt-5.4-nano	$+0.039$	$+0.144^*$	$+0.232^{***}$	$+0.068$
claude-sonnet-4.5	$+0.046$	$+0.135^*$	$+0.129$	$+0.061$
deepseek-v4-flash	$+0.133^*$	-0.072	-0.052	$+0.325^*$
gemini-2.5-flash	-0.005	$+0.015$	$+0.013$	-0.017

Table 8: Stage 2 across routed configurations. Spearman $\rho_S(u_0, \widehat{M}^{\text{Brier}})$ per (D, T). * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. 17 of 20 cells significant.

model	$D{=}2, T{=}1$	$D{=}2, T{=}2$	$D{=}3, T{=}2$	$D{=}5, T{=}2$
gpt-5.4-mini	$+0.180^{**}$	$+0.303^{***}$	$+0.219^{**}$	$+0.094$
gpt-5.4-nano	$+0.078$	$+0.220^{**}$	$+0.219^{**}$	$+0.123$
claude-sonnet-4.5	$+0.143^*$	$+0.178^{**}$	$+0.335^{***}$	$+0.253^{***}$
deepseek-v4-flash	$+0.231^{***}$	$+0.211^{**}$	$+0.169^*$	$+0.617^{***}$
gemini-2.5-flash	$+0.220^*$	$+0.207^{**}$	$+0.175^{**}$	$+0.156$

Table 9: Stage 1 (tokens) across routed configurations. Spearman $\rho_S(u_0, \text{completion tokens}_r)$ per (D, T). * $p < 0.05$, ** $p < 0.01$.

model	$D{=}2, T{=}1$	$D{=}2, T{=}2$	$D{=}3, T{=}2$	$D{=}5, T{=}2$
gpt-5.4-mini	$+0.087$	$+0.209^{**}$	$+0.186^{**}$	$+0.094$
gpt-5.4-nano	$+0.028$	$+0.105$	$+0.205^{**}$	$+0.076$
claude-sonnet-4.5	$+0.037$	$+0.145^*$	$+0.126$	$+0.071$
deepseek-v4-flash	$+0.163^*$	-0.028	-0.000	$+0.367^*$
gemini-2.5-flash	-0.010	$+0.029$	$+0.008$	-0.001

F.2 AUDITED DELEGATION EXAMPLES

For further clarification, we show two audited delegation examples.

For *"Will Elon Musk be the world's richest person on December 31, 2025?"*, the direct run reports conflicting evidence and says, "I did not find a definitive December 31, 2025 Bloomberg ranking." It outputs `FALSE, 0.56`, corresponding to $p_0 = 0.44$, and is wrong. The routed run delegates to a billionaire-wealth analyst, who retrieves the missing comparison, approximately $459B for Musk versus $340B for Ellison on September 30, and to an analyst of the Bloomberg index methodology. It concludes that Musk has "a meaningful cushion," outputs $p_r = 0.66$, and is correct.

For *"Will Crystal Palace win the 2025–26 English Premier League?"*, the direct forecast is already confident at $p_0 = 0.02$. The routed run still invokes both sub-agents, but they mainly confirm that Palace is 13th and not a plausible title contender; the forecast moves only to $p_r = 0.01$.

Quantity	Musk	Crystal Palace
Direct uncertainty u_0	0.88	0.04
Direct $P(\text{TRUE})$	0.44 (wrong)	0.02 (correct)
Routed $P(\text{TRUE})$	0.66 (correct)	0.01 (correct)
Routed agent cost ($/q)	0.0240 (87th percentile)	0.0172 (4th percentile)
Routed completion tokens	3,427 (92nd percentile)	1,898 (4th percentile)
Realized Brier improvement	+0.198	+0.0003

The pair traces both stages on one question each. The uncertain question ($u_0 = 0.88$) is where the Orchestrator spends, at 87th percentile agent cost and 92nd percentile completion tokens, and the delegation turns a wrong direct forecast into a correct one ($+0.198$ Brier). The confident question ($u_0 = 0.04$) draws 4th percentile spend on both measures, and delegation only confirms the answer the direct run already had ($+0.0003$). Spending follows direct-baseline uncertainty, and so does the payoff to spending.

G DATASET CONSTRUCTION AND DETAILS

We curate the 228-question evaluation dataset from ForecastBench (Karger et al., 2025). This section documents:

- **Source and sourcing pipeline.** The 228 questions are sub-sampled from 12 ForecastBench question-set partitions dated `2025-10-26` through `2026-03-29`; the per-partition counts are listed in Table 10.
- **Curation constraints.** (i) Every question's `market_info_close_datetime` is strictly after the `gpt-5.4-nano` knowledge cutoff (∼September 2025), with resolution dates February 2026–January 2027; (ii) the resolved label distribution is exactly 114 True / 114 False. The original ForecastBench partitions are False-heavy, so during curation we balance the set by including polarity-flipped variants of some questions: when a source question resolves False, its negated proposition is included as a True-resolved question where the wording remains natural and the resolution is unambiguous; (iii) topical mix is preserved from the underlying ForecastBench partitions (Politics/Geopolitics 49, Technology 32, Sports 31, Economics/Finance 21, Entertainment 5, Science/Climate 3, Other/Mixed 87).
- **Reproducibility metadata.** We retain the source ForecastBench identifiers, partition labels, resolution dates, topics, ground-truth labels, and polarity-flip indicators in the released question file, and credit ForecastBench as the source benchmark.

Table 10: Per-partition counts of the curated 228-question benchmark. Partitions are the underlying ForecastBench question-set release dates; the curation jointly imposes balanced T/F at the dataset level and post-cutoff resolution at the question level.

Partition (date)	N	#True	#False
2025-10-26	34	20	14
2025-11-09	28	10	18
2025-11-23	18	7	11
2025-12-07	23	12	11
2025-12-21	21	6	15
2026-01-04	10	5	5
2026-01-18	8	3	5
2026-02-01	9	4	5
2026-02-15	10	5	5
2026-03-01	24	16	8
2026-03-15	20	13	7
2026-03-29	23	13	10
Total	**228**	**114**	**114**

H SIDE EXPERIMENT: SENSITIVITY TO THE PRE-CUTOFF REGIME

The headline study (Section 4) deliberately uses post-knowledge-cutoff questions and a contamination-filtering web layer because pre-cutoff data is at risk of leakage through training-time exposure or through unfiltered retrieval of post-resolution content (Paleka et al., 2025). The severity of this risk is confirmed by Li et al. (2026), who show across 477 competition-level questions and 9 models that prompting an LLM to "ignore" knowledge acquired before its cutoff fails systematically: a 52% performance gap persists between simulated ignorance and true ignorance, and chain-of-thought reasoning *cannot* suppress prior knowledge even when reasoning traces contain no explicit post-cutoff references. This finding directly motivates the design of our pre-cutoff side experiment: we run it not to *use* pre-cutoff data as a fair test, but to diagnose whether the architecture-level allocation pattern persists under a different contamination regime. We therefore focus on qualitative system behavior, namely which architectures spend compute, call tools, and sit on the cost–quality frontier, rather than interpreting absolute accuracy as forecasting skill. A reviewer could reasonably worry that the *architectural* ranking we report, where orchestrator Pareto-dominates breadth and depth, is itself an artifact of how the post-cutoff + filtered setting interacts with each architecture. This side experiment is designed to falsify that worry.

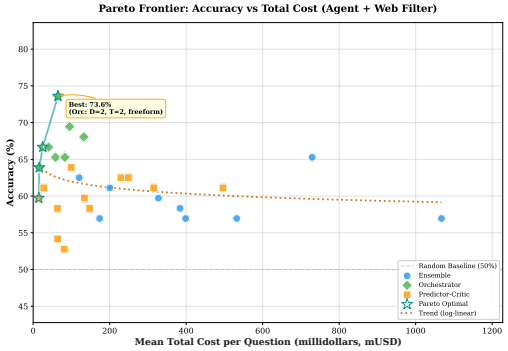
H.1 SETUP

We re-run the *exact* same code base and configuration sweep used in the main study, on a separate set of 72 ForecastBench questions dated `2024-12-08` whose resolution dates are *before* the model knowledge cutoff. The base model (`gpt-5.4-nano`), tools, web filter, midpoint cutoff convention, and analysis pipeline are identical; only the question set differs. This isolates the effect of the contamination-control regime on the architectural ranking and on each architecture's scaling behavior.

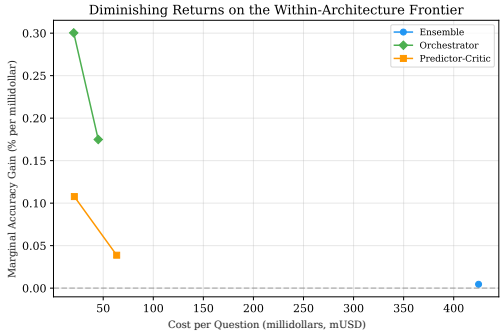
H.2 WHAT WE TEST

1. **Architectural ranking preservation**: does the orchestrator still Pareto-dominate ensemble and predictor–critic on cost, tokens, and LLM calls?
2. **Diminishing-returns shape preservation**: does the marginal-accuracy-per-extra-dollar curve along each architecture's efficient frontier still drop sharply after one or two points?
3. **Retrieval-use persistence**: do agents still spend test-time compute on tool use and retrieval even when questions resolve before the model cutoff? The web-filter cost share is a proxy for this behavior because it is incurred when agents retrieve pages that must be screened.
4. **Calibration profile**: do all architectures still exhibit non-trivial ECE, and is the headline post-hoc recovery procedure (Platt or temperature scaling) still effective in the contaminated regime?

H.3 RESULTS

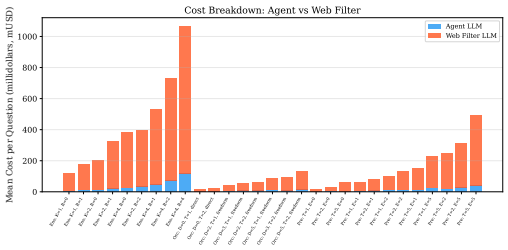


(a) Pareto: accuracy vs. total cost (72-Q pre-cutoff)

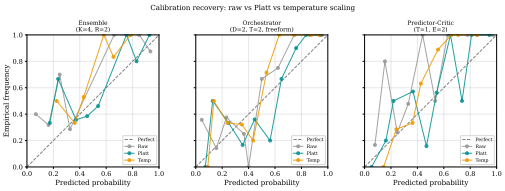


(b) Diminishing returns along the efficient frontier (72-Q)

Figure 10: Side experiment: architectural ranking and the diminishing-returns shape are preserved on the 72-question pre-cutoff set; the headline finding is not specific to the post-cutoff regime.

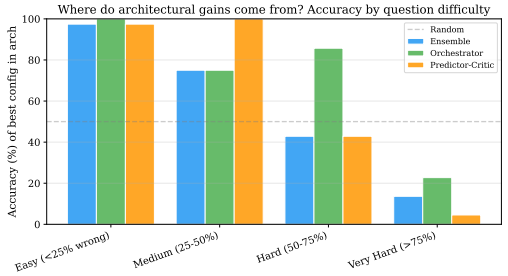


(a) Cost decomposition (72-Q pre-cutoff)

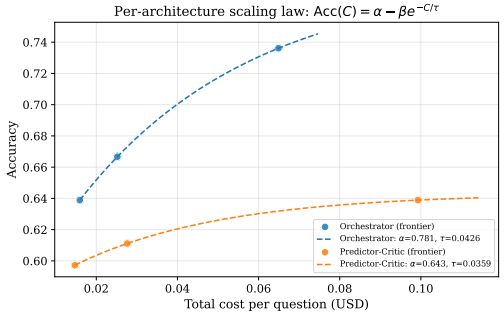


(b) Calibration recovery: raw vs. Platt vs. temperature (72-Q)

Figure 11: Agents continue to engage the retrieval pipeline on pre-cutoff questions: the web-filter cost share at the best configuration of each architecture remains high (89.2% Orch, 88.9% PC, 90.1% Ens). The side-experiment raw ECE values are higher than on the headline benchmark (0.17–0.23 vs 0.09–0.16), and post-hoc calibration is more variable: Platt scaling helps Ensemble and Orchestrator but hurts Predictor–Critic, while temperature scaling is more reliable across all three architectures.



(a) Difficulty stratification (72-Q)



(b) Saturating scaling-law fits (72-Q)

Figure 12: Difficulty stratification and the saturating scaling-law fit on the 72-Q side experiment.

The four checks above are preserved on the side benchmark (Figures 10 to 12), and the architectural ranking is in fact *stronger* on the pre-cutoff set than on the headline set:

- *Pareto ranking.* The orchestrator owns the entire cost–accuracy frontier on the side experiment. Best configurations: Orchestrator $D{=}2, T{=}2$ (freeform) at 73.6% **/ $0.065/q**;

Ensemble $K{=}4$, $R{=}2$ at 65.3% / \$0.729/q; Predictor–Critic $T{=}1$, $E{=}2$ at 63.9% / \$0.099/q. Same ordering, larger raw gap ($\sim$ 8–10 percentage points to either static policy).

- *Diminishing returns.* The marginal accuracy gain per extra dollar again drops by more than an order of magnitude after the first one or two frontier points, matching the headline shape.
- *Retrieval-use persistence.* The contamination-filter cost share at the best configuration of each architecture is 89.2% (Orchestrator), 88.9% (Predictor–Critic), and 90.1% (Ensemble); across the full 29-config side sweep the share spans 85.7–95.2% (median 90.7%). This is not important because the filter share is intrinsically interesting; rather, it shows that agents still make tool calls and engage the retrieval pipeline even in a regime where some outcome-relevant information may already be present in model weights.
- *Calibration is non-trivial but qualitatively different in direction.* All best-per-architecture configurations exhibit non-trivial raw ECE: 0.229 (Ens), 0.174 (Orch), and 0.170 (PC), substantially higher than the headline $[0.09, 0.16]$ band, consistent with residual training-time leakage that biases the model toward confident-but-not-always-right answers on pre-cutoff questions. Post-hoc recovery is also more variable than on the headline: Platt scaling reduces ECE by 60.9% on Ensemble and 16.2% on Orchestrator, but *worsens* ECE by 5.4% on Predictor–Critic; temperature scaling is the more reliable recovery method on the side benchmark (53.0% / 12.7% / 25.3% reduction across the three architectures). We treat this as expected: the calibration-recovery method best-matched to the paper's headline regime (two-parameter Platt on under-confident forecasts) is not necessarily best-matched to the contaminated regime where the bias structure of the raw forecasts is different. The calibration profile is therefore qualitatively non-trivial in both regimes but should be analyzed regime-by-regime, not assumed to transfer.
- *Statistical significance.* As on the headline study, bootstrap CIs straddle zero at $N{=}72$ (Orch vs. Ens: $\Delta\text{Acc} = +0.083, p = 0.11$; Orch vs. PC: $\Delta\text{Acc} = +0.097, p = 0.07$). The robust claim remains cost–quality dominance.

The slightly larger architectural gap on the side benchmark is consistent with the residual training-time leakage diagnosed by Li et al. (2026); Lahib et al. (2026): the orchestrator's ability to redirect retrieval evidence selectively may be even more useful when the underlying retrieval surface and parametric memory are partially contaminated. The validity claim the side experiment therefore supports is the *qualitative* preservation of the architecture-level allocation and retrieval-use patterns reported in the headline study, not a quantitative replacement of the headline numbers.

H.4 Where the side experiment differs

Two differences from the main study are expected and observed:

1. **Absolute accuracy is higher on the pre-cutoff set** for every architecture. We do *not* treat this as a positive result for the model: it is consistent with residual leakage through training-time exposure that the web filter cannot block, and is one of the reasons the headline study uses post-cutoff questions. Pre-cutoff exposure does not imply perfect accuracy: outcome-relevant facts may be absent from training data, not reliably retrieved from model weights, obscured by question wording or polarity flips, or blocked from retrieval by the web filter. We therefore interpret higher pre-cutoff accuracy as evidence of possible exposure, not as evidence that the task is trivial.
2. **Statistical power is lower** ($N{=}72$ vs. $N{=}228$); the side experiment is intended to validate *patterns*, not to provide a competitive accuracy number.

H.5 Take-away

The pre-cutoff experiment provides a sensitivity check rather than an independent forecasting evaluation. The qualitative cost–quality ranking and diminishing-returns pattern persist, but absolute accuracy may be inflated by training-time exposure, and the small sample does not establish statistically significant accuracy differences.

I FULL PER-CONFIGURATION TABLES

This appendix reports the full configuration grids underlying Table 1. We include accuracy, Brier score, total pipeline cost, and agent-side LLM calls; the released `metrics_by_config.csv` files include additional token, agent-only cost, web-filter cost, and utilisation columns.

Table 11: Full corrected `gpt-5.4-mini` configuration results.

Architecture	Config	N	Acc. (%)	Brier	Cost ($/q)
Ensemble	K=1, R=1	228	73.7	0.193	0.6922
Ensemble	K=1, R=2	228	78.1	0.180	0.8970
Ensemble	K=1, R=4	228	75.4	0.178	1.2412
Ensemble	K=4, R=1	228	76.8	0.181	1.9585
Ensemble	K=4, R=2	228	78.1	0.178	2.6631
Ensemble	K=4, R=4	226	77.9	0.172	3.6091
Orchestrator	D=0, T=1 direct	228	73.7	0.199	0.0469
Orchestrator	D=0, T=2 direct	228	76.3	0.178	0.0751
Orchestrator	D=2, T=1 freeform	228	77.6	0.183	0.1093
Orchestrator	D=3, T=1 freeform	228	74.6	0.207	0.1432
Orchestrator	D=2, T=2 freeform	228	80.7	0.153	0.1752
Orchestrator	D=5, T=1 freeform	228	75.4	0.194	0.2016
Orchestrator	D=3, T=2 freeform	228	75.4	0.211	0.2606
Orchestrator	D=5, T=2 freeform	228	78.1	0.190	0.3357
Predictor–Critic	T=2, E=0	228	64.5	0.224	0.0971
Predictor–Critic	T=5, E=0	228	68.9	0.201	0.2106
Predictor–Critic	T=2, E=1	228	71.1	0.199	0.2826
Predictor–Critic	T=2, E=2	228	72.4	0.185	0.4736
Predictor–Critic	T=5, E=1	228	72.8	0.195	0.5323
Predictor–Critic	T=5, E=2	228	75.4	0.169	0.7708
Predictor–Critic	T=2, E=5	228	76.3	0.169	0.9398
Predictor–Critic	T=5, E=5	228	76.8	0.168	1.6649

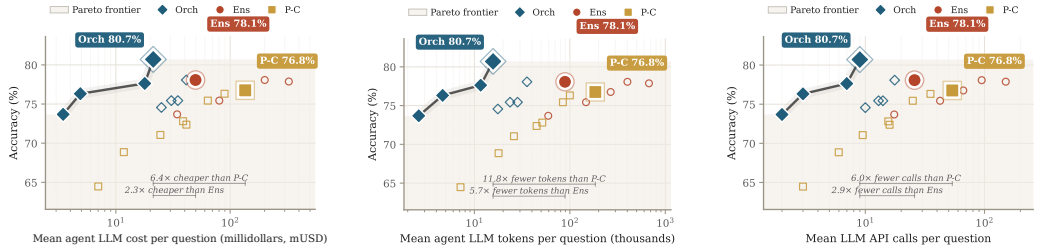
Table 12: Full `gpt-5.4-nano` configuration results.

Architecture	Config	N	Acc. (%)	Brier	Cost ($/q)
Ensemble	K=1, R=1	228	67.5	0.211	0.1566
Ensemble	K=1, R=2	228	68.4	0.193	0.2027
Ensemble	K=1, R=4	228	70.2	0.196	0.2942
Ensemble	K=4, R=1	228	70.2	0.195	0.4620
Ensemble	K=4, R=2	228	72.4	0.183	0.6086
Ensemble	K=4, R=4	228	72.8	0.179	0.8992
Orchestrator	D=0, T=1 direct	228	64.0	0.222	0.0111
Orchestrator	D=0, T=2 direct	228	68.9	0.209	0.0161
Orchestrator	D=2, T=1 freeform	228	64.9	0.226	0.0314
Orchestrator	D=3, T=1 freeform	228	65.4	0.224	0.0439
Orchestrator	D=2, T=2 freeform	228	68.4	0.221	0.0494
Orchestrator	D=5, T=1 freeform	228	68.9	0.218	0.0628
Orchestrator	D=3, T=2 freeform	228	71.1	0.209	0.0699
Orchestrator	D=5, T=2 freeform	228	70.2	0.211	0.1075
Predictor–Critic	T=1, E=0	228	61.0	0.234	0.0122
Predictor–Critic	T=2, E=0	228	63.6	0.236	0.0246
Predictor–Critic	T=1, E=1	228	65.4	0.230	0.0471
Predictor–Critic	T=5, E=0	228	63.2	0.225	0.0475
Predictor–Critic	T=2, E=1	228	63.6	0.218	0.0695
Predictor–Critic	T=1, E=2	228	63.6	0.230	0.0817
Predictor–Critic	T=2, E=2	228	64.0	0.212	0.1172
Predictor–Critic	T=5, E=1	228	68.9	0.207	0.1248
Predictor–Critic	T=1, E=5	228	68.0	0.210	0.1803
Predictor–Critic	T=5, E=2	228	71.5	0.194	0.1894
Predictor–Critic	T=2, E=5	228	70.2	0.202	0.2577
Predictor–Critic	T=5, E=5	228	68.9	0.209	0.4027

J ADDITIONAL PARETO VIEWS

The main results use total pipeline cost because the web-filtering layer is part of the contamination-controlled backtesting pipeline (Figure 2). To verify that the architecture ranking is not an artifact

of web-filter pricing, Figure 13 repeats the Pareto comparison using three agent-side or pricing-independent proxies: agent-only LLM cost, agent-token consumption, and LLM API calls. In all three views, the observed Pareto frontier is again composed of Orchestrator configurations only. Thus H1 is supported both in the full backtesting-cost accounting and in the agent-only compute accounting.

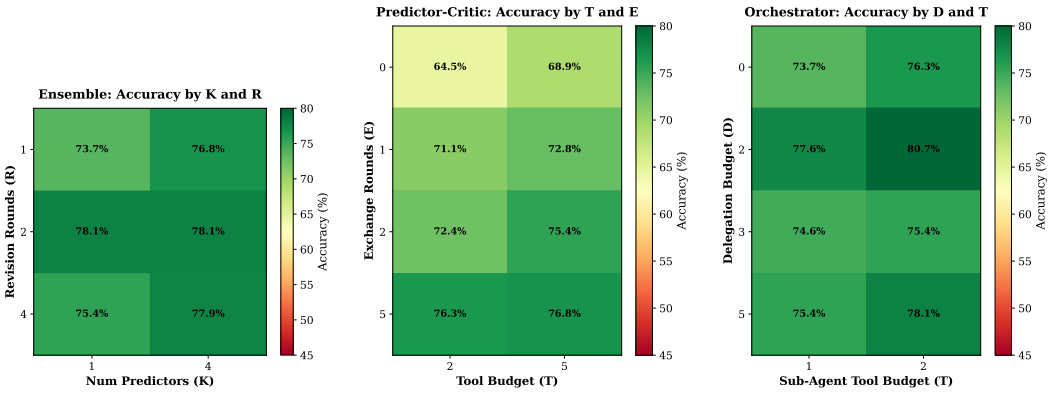


(a) Agent-only cost / question (b) Agent tokens / question (c) LLM API calls / question

Figure 13: Pareto views under alternative compute proxies. These reproduce the cost–accuracy ordering from Figure 2, showing that the result is not an artifact of web-filter pricing, dollar pricing, or token accounting.

K Full architecture-grid heatmaps

Figure 14 gives the full grid behind the saturation and non-monotonicity claim in Section 4.2. The heatmaps show that larger static schedules do not reliably improve accuracy. In the Ensemble grid, increasing from the cheapest $K{=}1, R{=}1$ setting to larger K, R can help, but the high-cost $K{=}4, R{=}4$ cell does not beat the cheaper $K{=}1, R{=}2$ or $K{=}4, R{=}2$ cells. In Predictor–Critic, increasing exchange depth and tool budget helps up to a point, after which additional compute yields small or non-monotone gains. In the Orchestrator grid, $D{=}2, T{=}2$ is the best cell; larger delegation budgets do not improve accuracy. This supports H2 in the empirical sense used in the main text: accuracy saturates or becomes non-monotone as budget grows, rather than increasing smoothly with raw test-time compute.



(a) Ensemble: $K \times R$ (b) Pred.–Critic: $T \times E$ (c) Orchestrator: $D \times T$

Figure 14: Accuracy heatmaps over each architecture's parameter grid. These provide full-grid transparency for the diminishing-returns result in Figure 3.

L Architecture-router oracle analysis

The best configuration of each architecture solves a partially different subset of questions. On the `mini` sweep, the best Ensemble ($K{=}1, R{=}2$) answers $178/228$ questions correctly, the best Orches-

trator (D=2, T=2) answers $184/228$, and the best Predictor–Critic (T=5, E=5) answers $175/228$. An oracle that selects the correct architecture per question among these three best configurations would answer $212/228$ questions correctly (93.0%). Among the 228 questions, all three architectures are correct on 142, all three are wrong on 16, exactly one is correct on 29, and exactly two are correct on 41. The unique-correct contributions are 8 for Ensemble, 16 for Orchestrator, and 5 for Predictor–Critic.

Table 13: Cross-architecture complementarity among the best `mini` configurations. "Oracle" is an upper bound: it selects the correct architecture after observing the label.

Selector	Correct	Acc. (%)	Notes
Best single config (Orchestrator)	184/228	80.7	D=2, T=2
Oracle over best three architectures	212/228	93.0	Uses realized labels
Oracle over all Ensemble configs	208/228	91.2	All K, R cells
Oracle over all Orchestrator configs	214/228	93.9	All D, T cells
Oracle over all Predictor–Critic configs	206/228	90.4	All T, E cells
Oracle over all reported `mini` configs	223/228	97.8	All reported cells

M CALIBRATION AND AGREEMENT DETAILS

Proper scores and calibration recovery. Table 14 reports accuracy, Brier score, log loss, and expected calibration error (ECE) for the best-accuracy configuration of each architecture in both model sweeps. Lower is better for Brier score and log loss. Calibration uses stratified 5-fold cross-fitted Platt scaling applied to logit $P(\mathrm{TRUE})$; log loss uses natural logarithms. ECE uses ten equal-width probability bins and is a calibration diagnostic rather than a proper score. The architecture with the highest accuracy is also the architecture with the best raw and calibrated proper scores in each sweep: Orchestrator for `mini` and Ensemble for `nano`. Accuracy and the proper scores provide complementary views: accuracy records whether the binary decision is correct, whereas Brier score and log loss evaluate the full reported probability and penalize confident errors. The complete `mini` ordering is not identical across metrics: Ensemble has higher accuracy than Predictor–Critic, whereas Predictor–Critic has lower Brier score and log loss. On `nano`, the Ensemble–Orchestrator accuracy gap is not statistically significant (exact McNemar $p = 0.68$), while the Ensemble uses $10.34\times$ the Orchestrator's agent-only cost. After Platt scaling, ECE is lowest for the Orchestrator in both sweeps. Given the 228-question sample and the post-hoc cross-fitted calibration, these differences should be interpreted cautiously.

Table 14: Accuracy and probabilistic scores for the best-accuracy configuration of each architecture. Accuracy uses the emitted binary label. Calibrated columns use stratified 5-fold cross-fitted Platt scaling. Lower is better for Brier score, log loss, and ECE.

				Brier score		Log loss		ECE	
Model	Architecture	Configuration	Accuracy (%)	Raw	Platt	Raw	Platt	Raw	Platt
`mini`	Predictor–Critic	T=5, E=5	76.8	0.168	0.152	0.527	0.482	0.129	0.092
`mini`	Ensemble	K=1, R=2	78.1	0.180	0.166	0.568	0.500	0.143	0.108
`mini`	Orchestrator	D=2, T=2	**80.7**	**0.153**	**0.141**	**0.479**	**0.454**	**0.119**	**0.081**
`nano`	Predictor–Critic	T=5, E=2	71.5	0.194	0.180	0.573	0.534	0.111	0.071
`nano`	Ensemble	K=4, R=4	**72.8**	**0.179**	**0.164**	**0.534**	**0.495**	0.123	0.048
`nano`	Orchestrator	D=3, T=2	71.1	0.209	0.193	0.607	0.564	0.133	**0.040**

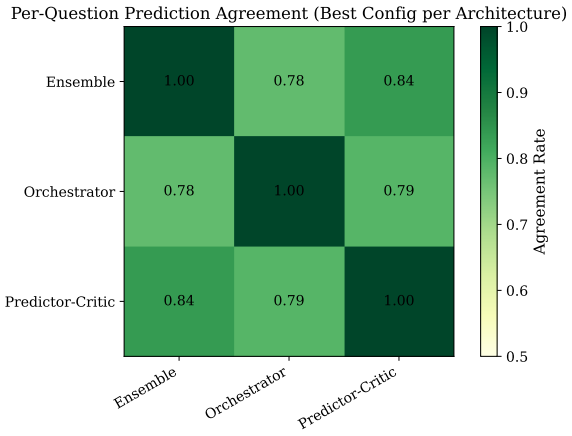
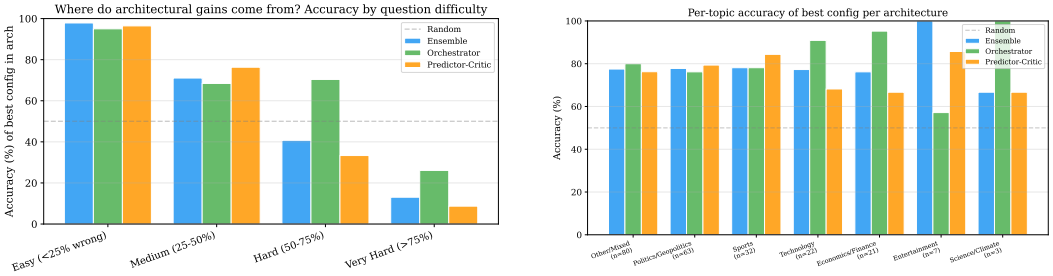


Figure 15: Pairwise binary-prediction agreement between the best-per-architecture configurations on the 228-question benchmark. Off-diagonal entries lie in $[77.2\%, 79.4\%]$, meaning the three architectures disagree on roughly 20% of questions. This is the headroom targeted by the architecture-routing direction discussed in Section 6.

N Difficulty and Topic Stratification

This section presents performance by question difficulty and topic.



(a) Accuracy by difficulty band

(b) Accuracy by topic

Figure 16: Where architectures differ across question strata. Difficulty bands are defined by across-architecture consensus error rate; topic labels come from the benchmark metadata.

O Theoretical-Framework Details

O.1 Formal Statement and Proof of Proposition 1

Setup. Fix a question–label distribution $(Q, Y) \sim \mathcal{D}$, a finite action set $\mathcal{A}$, a per-action unit cost $c : \mathcal{A} \to \mathbb{R}_{\geq 0}$, and a budget $C \geq 0$. A *compute-bounded policy* is a (possibly randomized) decision rule π that selects $a_t \in \mathcal{A} \cup \{\text{TERMINATE}\}$ from (Q, h_t), with $h_t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ the realized history of actions and observations, and that respects $\sum_t c(a_t) \leq C$ almost surely. On termination π emits $(\hat{y}, \hat{p}) \in \{0, 1\} \times [0, 1]$. Let $\Pi_{\text{static}} = \{\pi : a_t \perp h_t \mid Q, t\}$ and Π_{adapt} be all such policies. Let $L(\pi, C) = \Pr_{(Q,Y)\sim\mathcal{D}}[\hat{y} \neq Y]$.

Proposition (Restated). *For all $C \geq 0$, $\min_{\pi \in \Pi_{adapt}} L(\pi, C) \leq \min_{\pi \in \Pi_{static}} L(\pi, C)$. The inequality is strict whenever there exists a measurable event $E \subset \Omega$ with $\Pr[E] > 0$ on which the optimal Bayes action under $h_t \in E$ differs from the optimal Bayes action under $h_t \notin E$.*

Proof. Containment is immediate: $\Pi_{\text{static}} \subset \Pi_{\text{adapt}}$, so the minimum over the larger class is at least as small.

For strictness, let π_s^* achieve the static minimum and let $a_t^*(h_t)$ denote the Bayes-optimal action at step t given h_t. On E the static policy must, by definition, take an action independent of h_t, hence with positive probability strictly suboptimal under the 0–1 loss. Construct π_a as: identical to π_s^* off E, and equal to $a_t^*(h_t)$ on E. By construction π_a uses no more budget than π_s^* in expectation ($c(a_t^*) \le c(a_t^{\pi_s^*})$ by optimality of a_t^*); and $\Pr[\hat{y}_{\pi_a} \neq Y \mid E] < \Pr[\hat{y}_{\pi_s^*} \neq Y \mid E]$ by Bayes-optimality. Marginalising over E vs. $\bar{E}$ yields $L(\pi_a, C) < L(\pi_s^*, C)$. □

Remarks. (i) The proof is a one-step policy-improvement argument and does not require any specific form for the action distribution. (ii) The strict-inequality condition is mild: it asks only that the realized history sometimes carries information about the optimal next action, which is precisely what the orchestrator's freeform mode is designed to exploit. (iii) The proposition does not give a rate; the empirical rate is what the τ_{arch} in (4) measures.

O.2 Bayes-risk derivation of the evidence ceiling

We use the standard binary Bayes-risk bound rather than Fano's inequality. For binary Y, Fano gives $H(P_e) + P_e \log(|\mathcal{Y}| - 1) \ge H(Y \mid \hat{y})$, which with $|\mathcal{Y}| = 2$ collapses to $H(P_e) \ge H(Y \mid \hat{y})$. The further bound $H(P_e) \le 1$ yields $P_e \ge H(Y \mid \hat{y}) - 1$, but this is non-positive whenever $H(Y \mid R(C)) \le 1$, which is always the case for a binary outcome, so the bound is vacuous in our setting. We therefore use the (sharper, valid, and standard) decision-theoretic bound instead.

Let $\eta(R) = \Pr(Y = 1 \mid R)$. The minimum probability of error of *any* decision rule that bases $\hat{y}$ on the evidence set R is the Bayes risk $L^*(R) = \mathbb{E}_R\big[\min(\eta(R), 1 - \eta(R))\big]$, and any policy π acting on R satisfies $\Pr(\hat{y} \neq Y) \ge L^*(R)$. With $R = R(C)$, we obtain $L_{\text{ev}}(C) \ge L^*(R(C))$, the bound used in Appendix E.4. The implication that matters for the paper's argument is qualitative: additional reasoning compute on a fixed evidence set cannot reduce $L^*(R(C))$; only compute that changes the evidence (more retrieval, broader queries) can lower the floor.

O.3 Scaling-law fit procedure

We fit (4) to each architecture's efficient frontier using non-linear least squares with the Levenberg–Marquardt algorithm. The fitting target is mean accuracy across the 228-question benchmark, the regressor is total cost per question (USD), and we restrict to non-dominated frontier points to avoid contamination by clearly-suboptimal configurations. Confidence intervals are computed by a non-parametric bootstrap (1 000 resamples of frontier points; 2.5–97.5 percentile interval). The fitting script produces Table 15 and Figure 17.

Why the parametric form. The saturating exponential is the unique form implied by a first-order relaxation $d\,\text{Acc}/dC = (\alpha - \text{Acc}(C))/\tau$, i.e. each extra dollar of compute closes a fixed fraction of the remaining headroom to the evidence ceiling. The bounds $(\alpha, \beta, \tau) \in [0, 1] \times [0, 1] \times [10^{-6}, 10^{6}]$ make the fitted curve automatically monotone in cost and concave (diminishing returns), so neither has to be enforced post hoc.

Why the frontier filter. Restricting to non-dominated frontier points aligns the fitting data with these structural assumptions: a dominated configuration violates monotonicity by definition (lower accuracy at higher cost) and would otherwise force the optimiser to compromise between mutually inconsistent constraints. Empirically, 3/8 orchestrator configurations and 5/12 predictor–critic configurations sit on the Pareto frontier; all 5/5 ensemble configurations do (the breadth and revision axes both monotonically increase cost and accuracy in our sweep).

Robustness checks. We report three robustness analyzes in Appendix P: (i) *all-configurations* least-squares fit (no frontier filter), which inflates τ_{Orch} by $\sim 50\times$ as expected when monotonicity-violating points are admitted; (ii) a *per-question Bernoulli MLE* that replaces the implicit Gaussian-on-aggregated-accuracies likelihood with the correct $\text{correct}_i \sim \text{Bernoulli}(\alpha - \beta e^{-c_i/\tau})$ observation model and recovers Hessian-based Wald 95 % CIs. This agrees with the frontier fit for ensemble (where the per-question signal is large) but reveals that predictor–critic's τ is weakly identified at the per-question level (saturation is supported only by the frontier-aggregated reading); (iii) replacing

the exponential saturation by a power-law $\alpha - \beta C^{-\gamma}$ and replacing total cost by total tokens or total LLM calls. The qualitative ordering $\tau_{\text{Orch}} \ll \tau_{\text{PC}} < \tau_{\text{Ens}}$ is preserved across all three sensitivities.

Table 15: Fitted parameters of $\text{Acc}(C) = \alpha - \beta e^{-C/\tau}$ per architecture on the `gpt-5.4-mini` efficient frontier, with non-parametric bootstrap 95% intervals. τ is in USD per question.

Architecture	α (ceiling)	β (headroom)	τ ($/q)
Predictor–Critic	0.768 [0.746, 0.857]	0.159 [0.120, 0.318]	0.330 [0.187, 1.36]
Ensemble	0.774 [0.768, 1.000]	1.000 [–, –]	0.199 [0.185, 1874]
Hierarchical Orchestrator	0.848 [0.784, 1.000]	0.156 [0.105, 0.279]	**0.132** [0.035, 0.492]

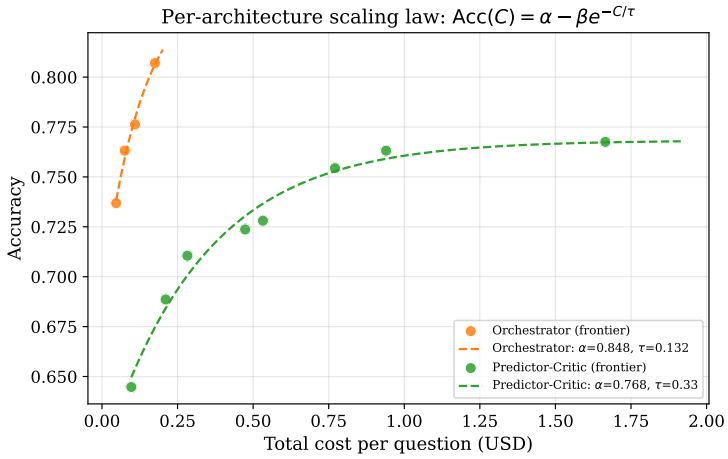


Figure 17: Per-architecture efficient frontiers with the fitted saturating model overlaid. The fits are descriptive: each frontier contains only a small number of non-dominated points and confidence intervals are correspondingly wide.

P Scaling-law robustness

We re-fit the saturating scaling law of Appendix O.3 under three perturbations of the fitting strategy and report the resulting (α, β, τ) in Table 16. The headline ordering $\tau_{\text{Orch}} \ll \tau_{\text{PC}} < \tau_{\text{Ens}}$ is preserved across all three checks; only the absolute scale of τ_{Orch} shifts when we admit dominated configurations into the fit.

Table 16: Three fits of $\text{Acc}(C) = \alpha - \beta e^{-C/\tau}$ per architecture on the 228-question balanced post-cutoff set (`gpt-5.4-mini`). *frontier-LSQ* is the headline procedure used in Table 15. *all-LSQ* drops the Pareto filter; *Bernoulli MLE* replaces the Gaussian-on-aggregated likelihood with an explicit per-question Bernoulli observation model.

Architecture	Mode	N_{obs}	α	β	τ ($/q)
Ensemble	all-LSQ	6	0.774	1.000	0.199
Ensemble	Bernoulli MLE	1366	0.774	1.000	0.198
Orchestrator	frontier-LSQ	4	0.848	0.156	0.132
Orchestrator	all-LSQ	8	0.770	0.641	0.016
Orchestrator	Bernoulli MLE	1824	0.770	0.644	0.016
Predictor-Critic	frontier-LSQ	8	0.768	0.159	0.330
Predictor-Critic	all-LSQ	8	0.768	0.159	0.330
Predictor-Critic	Bernoulli MLE	1824	0.769	0.159	0.336

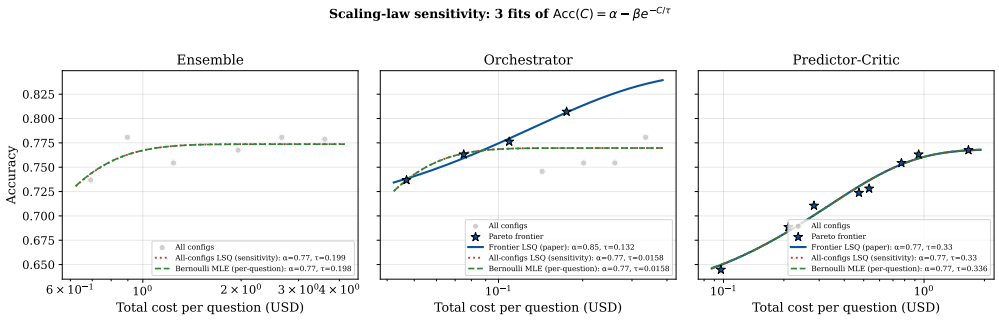


Figure 18: Sensitivity of the scaling-law fit. For each architecture we overlay the three fitting modes from Table 16. Light-gray markers are all observed configurations; navy stars mark the Pareto-optimal subset used by the headline fit. The headline (blue solid), all-configurations (red dotted), and Bernoulli-MLE (green dashed) curves agree closely for ensemble, confirming the architecture's frontier is essentially unaffected by fitting choice. They diverge for orchestrator: the all-configs and Bernoulli fits are pulled toward the dominated configurations and overestimate τ by roughly two orders of magnitude. Predictor-critic's Bernoulli fit collapses to a flat line because the per-question Bernoulli noise ($\approx$0.5) dominates the inter-config accuracy spread ($\approx$11pp); τ is therefore weakly identified at the per-question level even when it is well-determined on the frontier-aggregated curve.

Interpretation. For ensemble, every configuration is on the Pareto frontier and the per-question signal is modest, so the bounded exponential fit is best read as a compact descriptive curve rather than a parameter estimate. For orchestrator, including dominated configurations changes the fitted time constant sharply because those points lie at higher cost without higher accuracy. Predictor–Critic is more stable in the `mini` sweep because all complete configurations lie on the frontier used by the fit. We therefore report the frontier-LSQ values as headline descriptive summaries, accompanied by the explicit caveats above.

Q Experimental setup details

Models. All agents (predictor, critic, ensemble member, aggregator, sub-agents, and orchestrator) use either `gpt-5.4-nano` or `gpt-5.4-mini` via the OpenAI Responses API with reasoning effort `medium`; the web-filter LLM is held fixed. The main body reports the `mini` sweep as the primary complete architecture comparison, and uses `nano` as a full-sweep replication. We additionally report an Orchestrator-only cross-model check on complete delegation-enabled runs from Claude Sonnet 4.5, DeepSeek v4 Flash, Gemini 2.5 Flash, `gpt-5.4-mini`, and `gpt-5.4-nano` (Section 4.5).

Configuration sweep. For the main `mini` sweep, Predictor–Critic uses $T \in \{2, 5\}$ and $E \in \{0, 1, 2, 5\}$ (8 configs). Ensemble uses $K \in \{1, 4\}$ and $R \in \{1, 2, 4\}$ with the LLM aggregator (6 configs). Hierarchical Orchestrator uses $D \in \{0, 2, 3, 5\}$ and $T \in \{1, 2\}$, with `direct` mode for D=0 and `freeform` mode for $D \geq 2$ (8 configs). We apply a JSON-strict completeness filter that retains configurations with 228 per-question provenance JSON files; documented top-level API failures are excluded from the scored denominator. Thus the main `mini` analysis reports 22 configurations. Ensemble K=4, R=4 has 228 raw JSON files, but two top-level API failures, so it contributes 226 scored predictions. The fuller `nano` sweep, including the additional Predictor–Critic T=1 settings, is reported in the released metrics files and summary tables.

Baselines. We compare the three multi-agent architectures against each other in a controlled, like-for-like setup. No-tool single-pass, direct tool-use, and native reasoning-effort baselines are reported in Appendix T. Canonical self-consistency (Wang et al., 2023) with $N \in \{4, 8, 16\}$ remains future work.

Variance estimate. We report single-seed full-benchmark results and quantify uncertainty across questions using paired bootstrap intervals (Appendix R). Multi-seed variance remains a limitation

because of the cost of full provenance-logged reruns; we discuss this in Appendix W and design a small targeted seed study (best configuration per architecture, 50-question subset) as future work.

R STATISTICAL SIGNIFICANCE TESTS

The main results compare the best configuration of each architecture on the same 228 questions. Because the peak accuracies differ by at most 4.0 percentage points, we treat raw-accuracy ordering cautiously and use paired tests to quantify uncertainty. Table 17 reports paired bootstrap confidence intervals for accuracy differences and exact two-sided McNemar tests.

Table 17: Paired accuracy comparisons between the best configuration of each architecture on the same 228 questions. Bootstrap CI: paired 10 000-resample. McNemar: exact two-sided.

A	B	ΔAcc 95% CI	McNemar p
Ensemble K=1, R=2	Orchestrator D=2, T=2	$[-0.088, +0.035]$	0.49
Ensemble K=1, R=2	Pred.–Critic T=5, E=5	$[-0.039, +0.066]$	0.75
Orchestrator D=2, T=2	Pred.–Critic T=5, E=5	$[-0.018, +0.101]$	0.25

S CROSS-MODEL FULL RESULTS

The complete `gpt-5.4-mini` and `gpt-5.4-nano` sweeps are written to the released metrics files. For the quick five-model Orchestrator comparison in Section 4.5, we restrict to the selected delegation-enabled settings and require complete 228-JSON provenance. Table 18 reports the best row per model; the full file includes all 18 complete delegation-enabled Orchestrator configs.

Table 18: Best complete Orchestrator configuration per model among the selected delegation-enabled settings.

Model	Best delegation-enabled config	Correct	Acc. (%)	Cost ($/q)
`gpt-5.4-mini`	D=2, T=2 freeform	184/228	80.7	0.175
Claude Sonnet 4.5	D=2, T=2 freeform	183/228	80.3	0.912
DeepSeek v4 Flash	D=3, T=1 freeform	183/228	80.3	0.037
Gemini 2.5 Flash	D=3, T=2 freeform	163/228	71.5	0.165
`gpt-5.4-nano`	D=3, T=2 freeform	162/228	71.1	0.070

The full `nano` architecture sweep is useful as a robustness check rather than the main headline. Its best raw-accuracy configuration is Ensemble K=4, R=4 at $166/228 = 72.8\%$, while the best Orchestrator is D=3, T=2 at $162/228 = 71.1\%$. The paired accuracy difference is not statistically resolved: the bootstrap 95% CI for Ensemble minus Orchestrator is $[-4.4, +8.3]$ percentage points and McNemar's exact two-sided $p = 0.68$. The cost gap is large, however: the Orchestrator costs \$0.070/question versus \$0.899/question for the Ensemble (about $13\times$ cheaper). Thus the nano sweep supports the cost–quality conclusion even though its raw best point differs from the `mini` sweep.

T NO-TOOL AND DIRECT BASELINES

We added a zero-shot, single pass CoT with no tools baseline for both `mini` and `nano` models on all questions; the results are reported below.

Table 19: No-tool, zero-shot, single-pass baseline results.

Model	Accuracy	Raw / Platt Brier	Raw / Platt log loss
`mini`	68.4%	0.210 / 0.197	0.627 / 0.587
`nano`	62.7%	0.230 / 0.215	0.660 / 0.620

The selected best configuration of each of Predictor–Critic, Ensemble, and Orchestrator improves over the corresponding no-tool baseline.

Also, the existing $D{=}0, T{=}1$ Orchestrator configuration for each model, already reported in Tables 11 and 12, provides a second baseline: one retrieval call and no delegation. These baselines show the observed progression from no external evidence, through minimal direct tool use, to the selected routed Orchestrator configuration.

Table 20: Progression from no external evidence to direct tool use and the selected routed Orchestrator.

Model	No-tool $N{=}1$	Direct $D{=}0, T{=}1$	Selected routed	Direct minus no-tool	Routed minus direct
`mini`	68.4%	73.7%	80.7% ($D{=}2, T{=}2$)	+5.3 pp, $p{=}0.096$	+7.0 pp, $p{=}0.033$
`nano`	62.7%	64.0%	71.1% ($D{=}3, T{=}2$)	+1.3 pp, $p{=}0.780$	+7.0 pp, $p{=}0.023$

Thus, enabling one retrieval call alone does not account for the full observed gain. For both `mini` and `nano`, the selected routed Orchestrator outperforms the $D{=}0, T{=}1$ direct baseline by 7.0 percentage points. Going from no tools to one retrieval call helps `mini` more than `nano` (+5.3 points versus +1.3 points).

We also swept the models' own reasoning effort, with no tools, on all questions:

Table 21: No-tool native reasoning-effort sweep.

Model	Effort	Accuracy	Brier	Log loss
`mini`	low	73.2%	0.191	0.596
`mini`	medium	**74.6%**	**0.184**	**0.572**
`mini`	high	73.2%	0.190	0.591
`nano`	low	68.0%	0.219	**0.638**
`nano`	medium	71.1%	0.217	0.669
`nano`	high	**71.9%**	**0.215**	0.655

This is a strong baseline, and it saturates early too: `mini` peaks at medium effort and falls back at high. The best `mini` Orchestrator still reaches 80.7% against 74.6% here ($+6.1$ pp, McNemar $p{=}0.038$), though that comparison is post-hoc and not with matched compute. We plan to add the canonical self-consistency with independent samples at $N \in \{4, 8, 16\}$.

U METADATA-BASED CONTAMINATION-FILTER RECALL AUDIT

The web filter emits per-page pass/fail decisions during retrieval. To audit whether it catches obvious leakage, we construct metadata-based proxy labels from page dates, result snippets, and filter metadata. This is deliberately a proxy: it catches pages that look leakage-prone from metadata, not a human judgement of whether the page truly contaminates the forecast. Across 847431 page-level filter decisions, the broad proxy identifies 141159 likely leakage pages, of which 126563 are blocked, giving proxy recall 89.7% with Wilson 95% CI $[89.5, 89.8]\%$. Under a stricter proxy definition, 68301 pages are flagged and 62876 are blocked, giving proxy recall 92.1% with CI $[91.9, 92.3]\%$.

The audit also records candidate misses: pages retained by the filter but flagged by the proxy. These are useful for filter debugging. Because these labels are metadata based, the proxy audit should be interpreted separately from the semantic human-audit results reported in the main text.

V COST BREAKDOWN FULL RESULTS

Table 3 in the main text reports representative percentage splits between agent/model calls and web-filter calls. Figure 19 gives the full percentage view, and Figure 20 shows the same phenomenon in dollar units. The conclusion is unchanged: the web-filtering layer dominates total pipeline cost across the corrected `mini` sweep, while the Orchestrator configurations remain at the low-cost end because they retrieve less evidence and invoke fewer filtered pages.

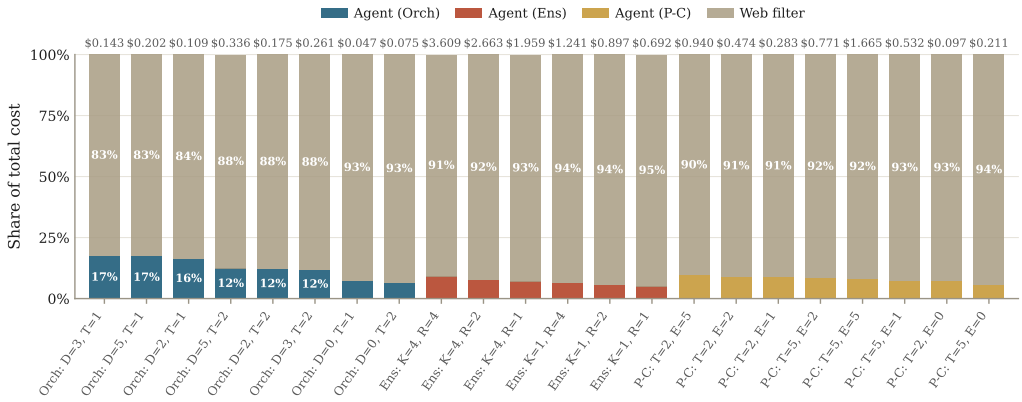


Figure 19: Full cost-share decomposition per configuration. The contamination filter accounts for 83–95% of total spend across the `mini` sweep, with the Orchestrator family at the lower end of that range.

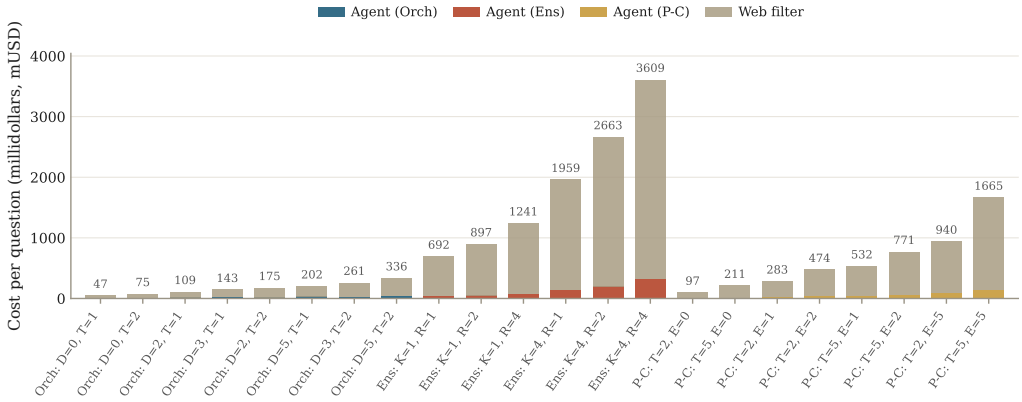


Figure 20: Cost decomposition per configuration. The contamination filter accounts for 83–95% of total spend across all `mini` configurations (median 91.8%); the Orchestrator family sits at the lower end of this range because its retrieval budget is history-conditioned.

This appendix view is useful for separating two facts that are compressed in the main-text summary table. First, the web filter is the majority of spend for every configuration. Second, the absolute dollar totals still vary substantially across architectures: static breadth and depth incur larger retrieval/filtering bills because their macro-schedules trigger more tool use, whereas the Orchestrator's history-conditioned routing keeps both agent-side and filter-side costs lower.

W LIMITATIONS

Our results should be interpreted as a controlled empirical study of compute allocation for agentic forecasting, not as a complete evaluation of all possible forecasting systems or deployment settings. The main limitations concern model and architecture coverage, statistical power, contamination-filter validation, benchmark domain balance, and the extent to which API costs proxy for underlying compute.

- **Partial non-OpenAI architecture coverage.** We have complete `gpt-5.4-nano` and `gpt-5.4-mini` architecture sweeps (Orchestrator, Ensemble, Predictor–Critic), plus complete subsets of Orchestrator runs for Claude Sonnet 4.5, DeepSeek v4 Flash, and Gemini 2.5 Flash. The five-model Orchestrator comparison in Section 4.5 should therefore

be read as a complete-config snapshot for selected delegation-enabled settings, not as a full-factorial model-family sweep; a full architecture × parameter × model sweep is left for future work.

- **Self-consistency baseline deferred.** Canonical self-consistency with independent samples at $N \in \{4, 8, 16\}$ remains future work. The within-paper architecture comparisons are unaffected by its absence.
- **Single seed per configuration.** We rely on $N{=}228$ questions for variance reduction rather than seed averaging. Multi-seed estimation on targeted subsets remains future work.
- **Statistical resolution.** At $N{=}228$ and the observed inter-architecture gap (≤ 4.0 percentage points in the `mini` sweep), the McNemar test is not powered to resolve accuracy differences among best configurations (Appendix R); the robust claim is cost–quality dominance, not a statistically decisive raw accuracy ordering.
- **Filter-audit coverage.** The human audit uses a blinded 200-page sample, so its recall, precision, and residual- contamination estimates remain subject to sampling uncertainty.
- **Domain coverage.** Topic distribution is set by ForecastBench; rare domains (Science/Climate $n{=}3$, Entertainment $n{=}5$) are underrepresented.
- **Tool surface and delegation mode.** The freeform orchestrator defines sub-agent roles at runtime; a predefined variant (fixed Research / Data / Analysis sub-agents) was implemented but not swept. Whether different fixed-role designs or hybrid predefined/freeform schedules would change the cost–accuracy tradeoff is an open question.
- **Cost-as-proxy.** USD pricing reflects current OpenAI rates; we therefore report token- and call-based proxies alongside USD throughout (Section 4).