ULTRALPHA: CONTINUAL ALPHA MINING AS RESEARCH COMPUTE SCALES

Anonymous authors
Paper under double-blind review

ABSTRACT

Mining alpha factors is an iterative research process. In practice, a quant researcher writes a hypothesis as factor code and backtests it, letting each outcome decide the next attempt. Large language model (LLM) agents now conduct autonomous research for days, and a growing number of works have them run this loop, writing and testing factor code to mine new factors. While these methods have proven effective at producing candidates, our systematic analysis finds that their output repeats known factors and decays quickly in future tests. To mitigate these problems, we propose ULTRALPHA, a closed-loop harness for long-horizon alpha research. It keeps the agent researching in one protected loop with metered feedback and recorded actions, and carries its progress in a persistent research state that deduplicates output and keeps verified evidence. We evaluate ULTRALPHA on the A-share CSI300 universe with three base models. ULTRALPHA keeps mining new alpha. Under the same frozen evaluation, after deduplication 32 to 46 of its delivered factors stay positive on the test window, a retention of 95.7% to 97.0% of all delivered factors. Its Top-10 average 24h development IC is 0.0536 to 0.0576, and that average stays positive on the test window in all three blocks. Traded factors keep compounding while baseline curves rise and fade. Further analysis shows strong generalization of the framework to minute-level crypto perpetual markets and alpha mining continuing as research compute scales.

1 INTRODUCTION

> *"The person who turns over the most rocks wins the game."*
>
> —*Peter Lynch* (Tanous, 1997)

Alpha factors, programs that turn market history into signals of future returns, are the basic unit of quantitative research (Kakushadze, 2016). In the real workflow, a quant researcher tests one factor after another against market history, and the score steers the next round of the loop. LLM agents now write code, run experiments, and revise their own plans over days of continuous work (Lu et al., 2024; Novikov et al., 2025; Jiang et al., 2025; Schmidgall et al., 2025). It is therefore natural to expect an agent to run the same mining loop autonomously. The question that matters is whether an agent can move beyond producing one plausible factor and keep producing discoveries that are genuine and mutually distinct, the way the best human researchers do.

Traditional methods mine factors by searching over formula trees for predictive expressions (Yu et al., 2023; Shi et al., 2025; Chen et al., 2026). Recently, a growing number of methods use LLMs to write and refine factor code, and they produce alpha factors at scale (Wang et al., 2025); QUANTAGENT follows a self-improving search loop (Wang et al., 2024), alongside recent systems such as ALPHAAGENT, RD-AGENT(Q), QUANTAALPHA, COGALPHA, and ALPHALOGICS (Tang et al., 2025; Li et al., 2025; Han et al., 2026; Liu et al., 2026b; Weng et al., 2026). Yet how well these methods sustain discovery when mining never stops has gone unmeasured. In Section 2, we first analyze several popular methods under one frozen panel and find that although they can often mine useful factors, their output reveals three limitations.

❶ **Development performance does not transfer.** Every candidate is scored on one evaluation window, and every repeated query fits that window a little better (White, 2000; Bailey & de Prado, 2014; Bailey et al., 2017). The development score keeps rising after the factor has stopped predicting.

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

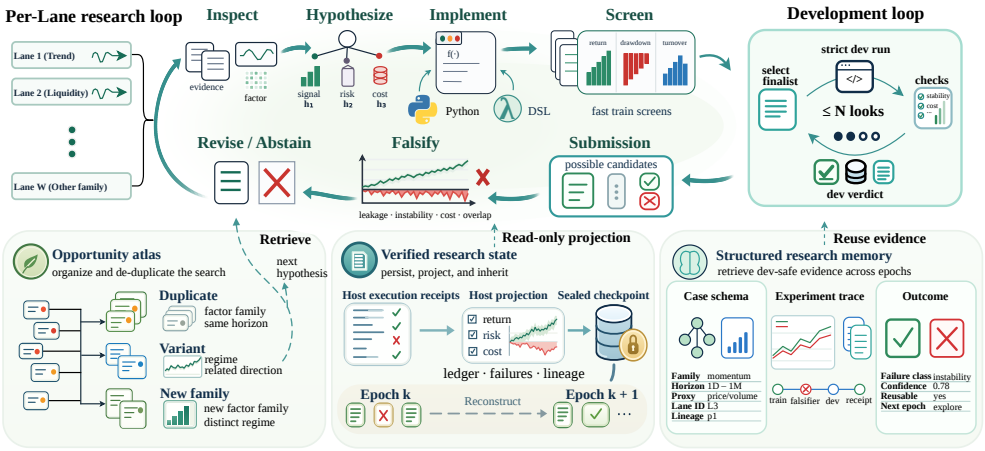

Figure 1: The ULTRALPHA method. Each lane inspects the panel, states a signal, a risk, and a cost, and writes the factor in Python or in a domain-specific language. A fast training screen reports return, drawdown, and turnover, and the development loop spends at most N looks before a verdict. The opportunity atlas, the verified research state, and the structured research memory then carry that search from one epoch into the next.

❷ A production count is not evidence. Progress is usually reported as a production count (Luo et al., 2026). But the share of produced candidates that stay positive in the test window varies widely across runs, so two runs with the same count can carry very different amounts of evidence. **❸ Production inflates through rewrites.** A factor that rewrites a known one can still backtest well even though the crowded signal no longer earns anything (McLean & Pontiff, 2016). As mining continues, production climbs while the supply of genuine mechanisms runs out.

To address these limitations, we propose ULTRALPHA, a closed-loop harness for long-horizon alpha research. Inside the loop, the agent works like a quant researcher, writing its own analysis code and turning hypotheses into factors across parallel lanes. Across epochs, a persistent research state and an opportunity atlas remember what has been tried and verified, so the search builds on its past instead of repeating it. Outside the loop, development evaluations are metered and the test judge stays hidden, so the search cannot spend the evidence that certifies discovery. We evaluate ULTRALPHA on the A-share CSI300 universe with three base models. ULTRALPHA keeps discovering new factors. Under the same frozen evaluation, 46, 32, and 45 of its delivered factors stay positive on the test window after deduplication, 95.8% to 97.0% of each delivered set. Its top-10 average 24h development IC is 0.0536 to 0.0576, and that average stays positive on the test window in all three blocks. Traded factors keep compounding while baseline curves rise and fade. Further analysis shows strong generalization to minute-level crypto perpetual markets and alpha discovery continuing as research compute scales. Our contributions are as follows.

- We analyze popular mining methods systematically under one frozen panel and surprisingly discover that although useful factors emerge, turning sustained search into genuine discovery remains difficult, no matter how long the search runs.
- To mitigate these limitations, we propose ULTRALPHA, a closed-loop harness for long-horizon alpha research that keeps an LLM agent researching in one protected loop and carries verified, deduplicated research state across epochs.
- We report CSI300 comparisons across three base models under one frozen evaluation. After deduplication, 46, 32, and 45 of its delivered factors stay positive on the test window, 95.7% to 97.0% of each delivered set, and its top-10 average 24h development IC reaches 0.0576. Further analysis shows strong generalization of the framework to minute-level crypto perpetual markets.

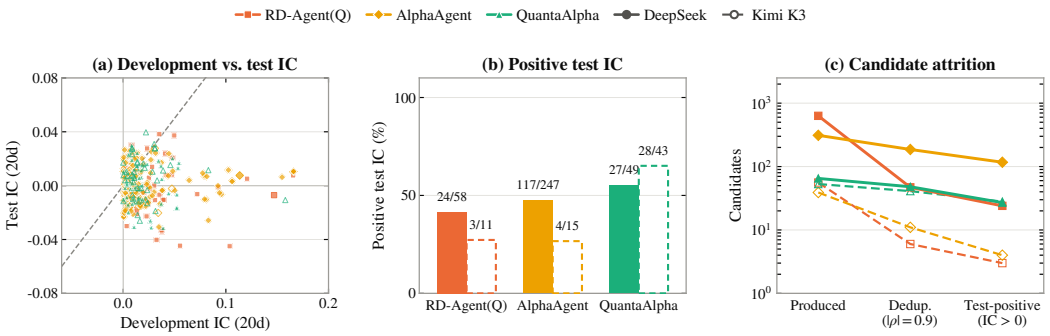

Figure 2: A-share CSI300 diagnostics of three native mining methods at the 480h horizon. Colors identify methods, with solid lines for DEEPSEEK-V4.1-FLASH and dashed lines identify KIMI-K3. Development performance does not transfer reliably to test. Only 27% to 65% of produced candidates stay positive. Deduplication removes 23% to 93% of production.

2 FORMULATION AND DIAGNOSTICS

2.1 PROBLEM FORMULATION

An alpha factor is a program that reads market history and assigns each asset a score, and it is useful exactly when its ordering of assets predicts future returns. Mining such factors is iterative, and each backtest on historical data decides the next attempt. Existing mining systems automate this loop, and they share one template. A candidate space $\mathcal{F}$ fixes what can be proposed, and a backtest evaluator $B(f; D)$ scores each proposal on historical data D. Iterating on the scores so far, the system reports the highest-scoring set after K iterations,

$$\hat{\mathcal{F}} \;=\; \mathrm{top}_m \left\{ f \in \{f_1, \ldots, f_K\} \,\middle|\, B(f; D) \right\}, \tag{1}$$

where top_m keeps the m highest-scoring candidates. The score that guides the search is visible to the searcher, so every repeated query fits the evaluation window a little better, and that access belongs inside the evaluation protocol.

2.2 FROZEN-PANEL DIAGNOSTICS

Experimental setup. We run the study on the A-share CSI300 point-in-time panel, with D_{train} from 2008 through 2021, D_{dev} from 2022 through 2024, and D_{test} from January 1, 2025 through September 11, 2026. The chosen methods are RD-AGENT(Q) (Li et al., 2025), ALPHAAGENT (Tang et al., 2025), and QUANTAALPHA (Han et al., 2026), run with their default settings. Each one runs under two backbones, DEEPSEEK-V4.1-FLASH and KIMI-K3. We report the Rank IC, the Spearman correlation between factor scores and forward returns, averaged across sessions within each evaluation window (Kakushadze, 2016).

Findings. We present our results in Figure 2 and draw three findings. ❶ **Development performance does not transfer reliably.** In panel (a), candidates from every method fall below the equality line, including ones with positive development scores. A factor can therefore score well during selection and lose its predictive power on data the search never used. One window cannot both guide a search and certify its result, so a harness that runs for days must hold part of the data beyond the searcher's reach and re-evaluate candidates on it before counting them as alpha. ❷ **A production count is not evidence.** In panel (b), the share of produced candidates that stay positive in the test window ranges from 27% to 65% across the six runs, and some output cannot be replayed at all. Positive test IC is this share, and it measures directional persistence, not statistical significance or profitability after trading costs. Two runs with the same count can therefore carry very different amounts of evidence. The production count carries no evidence unless it is reported together with the number of candidates that can be replayed and the number that stay positive. ❸ **Production inflates through rewrites.** Deduplication removes a candidate that behaves like a kept one, with mean ranking correlation of magnitude at least 0.9 over at least 30 shared assets and 60 shared dates. In panel (c) it removes 23% to 93% of production, and one RD-AGENT(Q) run falls from 631 produced to 47 retained and 24

still positive, the two counts taken separately. The runs are not proposing 631 different ideas. They rewrite a small set of factors with tweaked constants and shifted windows. A search that runs without bound must remember what it has already found, or its count grows through rewrites of the same factor rather than new alpha.

3 THE ULTRALPHA METHOD

To address the above problems, we propose ULTRALPHA, a closed research loop for long-horizon alpha mining. The model writes each factor, the host returns only bounded evidence for the next decision, and a persistent state keeps the search from treating rewrites as new discoveries. Figure 1 shows the method.

3.1 THE RESEARCH LOOP

Research transition. A task τ fixes the point-in-time data view, the target horizons, the candidate interface, the costs, and the feedback policy. At step r, the model chooses an action a_r from the visible state s_r, its case memory m_r, and the opportunity atlas A_r, and the host executes that action and projects the ledger back into the model's view,

$$a_r = \pi(\tau, s_r, m_r, A_r), \qquad (L_{r+1}, o_r) = H_\tau(L_r, a_r), \qquad s_{r+1} = P(L_{r+1}). \tag{2}$$

Here L is the host ledger, H_τ is the tool interface, o_r is bounded feedback, and P is a read-only projection. The model sees that projection rather than the raw receipts, so it cannot replace the host's record with its own recollection of what happened.

Loop actions. The loop runs from inspection through revision. The model inspects the panel with its own analysis code and states a hypothesis that names a signal, a risk, and a cost, together with a competing explanation and a falsifier, so a later failure has a reason the search can check. It then writes the factor, and the host screens that factor on training data, reporting return, drawdown, and turnover while keeping the raw cell records. The model selects a finalist for a development look, and a verdict may admit possible candidates without yet counting them as discoveries. When the evidence shows leakage, instability, cost, or overlap, the host falsifies the candidate, and the model either revises it by naming the previous attempt and the change or abstains, with the failed attempts kept in the ledger for the return to inspection.

Two candidate languages. The factor is written in one of two languages under a single contract. The domain-specific language (DSL) follows the expression language of Qlib (Yang et al., 2020) and the formulaic-alpha tradition (Kakushadze, 2016), and it forms a closed algebra of trailing-window, cross-sectional, and elementwise operators, so $\mathrm{Mean}(\mathrm{close}, 20)$ satisfies $s_t = f(X_{\le t})$ by construction and cannot look ahead. Python is the other language, a stateful program that walks the panel row by row,

$$s_t = f(x_t, X_{<t}; \omega_t), \qquad \omega_{t+1} = g(\omega_t, x_t), \tag{3}$$

where x_t is the current cross-section, $X_{<t}$ is the trailing history, and ω_t is the program's internal state. Before execution, three static checks reject unsafe imports, lookahead, and dynamic field access. Both languages emit one factor panel, and a candidate $c = (h, f, d, \ell)$ binds the factor to its hypothesis, direction, and lineage, with the source hash of f as its identity.

Confirmation under budget. A development look is the strict check that follows screening. The model may spend at most N such looks in an epoch, and it decides when a look is worth spending rather than following a fixed schedule, which stops the search from fitting the development window by asking the same question again and again. A look on a factor the host has already seen replays the stored record and does not spend the budget again. The run checks stability and cost, and it also checks execution, predictive strength, and overlap, while controlling the false discovery rate across factor-horizon cells with a Benjamini-Hochberg rule (Benjamini & Hochberg, 1995), $p_{(i)} \le i\alpha/m$, together with per-cell thresholds. Every look enters the shared development ledger.

3.2 PERSISTENT RESEARCH STATE

ULTRALPHA continues across epochs, and each epoch starts a fresh thread, so the memory has to outlive the thread or the next epoch repeats the same search. Three stores carry that memory. An epoch closes with a verified projection $V_k = R(L_{\le k})$, and the next epoch opens with a brief $b_{k+1} = B(V_k, A_k)$.

Table 1: Delivered factors on A-share CSI 300 by backbone under frozen evaluation. ULTRALPHA leads every backbone block on retention, the development average, return, and drawdown. The best model in each column is in **bold**.

Backbone	Method	Kept (N)	Retention (%)	Dev IC	Test IC	ARR (%)	MDD (%)↓	Sharpe
Model-free (backbone-agnostic)								
	AlphaForge-AFF	5	**100**	0.0246	+0.0164	+7.2	−19	+0.59
	Vanilla GP	**52**	52.0	**0.0334**	**+0.0360**	−0.7	−49	+0.10
DEEPSEEK-V4.1-FLASH								
	ULTRALPHA	46	**95.8**	**0.0576**	**+0.0289**	**+6.2**	**−16**	**+0.49**
	RD-Agent	20	95.2	0.0231	+0.0175	+0.3	−30	+0.11
	AlphaAgent	**52**	74.3	0.0300	+0.0244	+2.2	−19	+0.23
	QuantaAlpha	12	80.0	0.0140	+0.0141	+0.3	−43	+0.13
KIMI-K3								
	ULTRALPHA	32	**97.0**	**0.0567**	+0.0227	**+8.8**	**−18**	**+0.71**
	RD-Agent	11	78.6	0.0185	+0.0133	+6.4	−33	+0.39
	AlphaAgent	**33**	76.7	0.0277	**+0.0243**	+2.2	−25	+0.23
	QuantaAlpha	17	73.9	0.0148	+0.0110	−7.4	−62	−0.13
GLM-5.3-FLASH								
	ULTRALPHA	**45**	**95.7**	**0.0536**	**+0.0406**	**+9.2**	**−18**	**+0.73**
	RD-Agent	7	87.5	0.0148	+0.0085	−0.4	−34	+0.08
	AlphaAgent	6	85.7	0.0158	+0.0237	−4.9	−40	−0.21
	QuantaAlpha	31	77.5	0.0402	+0.0130	−0.2	−22	+0.07

Structured research memory. A case is the model's own note of what it tried, recording the family, the horizon, the outcome, and the failure class. The host treats these notes as leads rather than evidence, because the model writes them, and retrieval can return only evidence that is safe to show during development, after references to protected outcomes have been removed. A later epoch inherits this memory read-only, beside the host projection.

Verified research state. This record is what the host can prove, not what the model claims. The host rebuilds it from execution receipts alone, because self-reported outcomes have disagreed with the ledger, and a candidate counts as passed only when the ledger holds no rejection for it, while a broken record can only demote it and never promote it. The read-only projection tells the model whether return, risk, and cost passed, together with the failures and the lineage, without handing over the raw measurements or spending a development look.

Opportunity atlas. The atlas keeps the search from rewriting a factor it has already found, and it assigns one of three labels. A duplicate is the same factor family at the same horizon, a variant keeps a related direction and changes the regime, and a new family is a new factor family in a distinct regime. The host tells these apart with source hashes, structural fingerprints, and quantile profiles of the factor panel. A profile cosine of at least 0.9 is a near copy rather than a new family, and a candidate with too little evidence stays unknown rather than new.

3.3 PARALLEL LANES

Research width adds concurrent lanes, and research depth adds epochs inside one lane, so the search can cover more families without starting over. Each lane runs its own copy of the loop in Section 3.1 on one pre-defined mechanism family, with its own notes, and writes into the shared state of Section 3.2. Lanes do not read each other's unfinished work, which keeps one lane from quietly copying another. A supervisor seals each lane's results under one lock and owns the development budget, so every look enters one shared ledger, and an interrupted lane is sealed before its records can be reused. At epoch closeout the host merges the ledgers,

$$V_k = R\Big(\bigcup_{w=1}^{W} L_{\le k}^{(w)}\Big), \tag{4}$$

where R rebuilds the verified research state and $L_{\le k}^{(w)}$ is the ledger of lane w through epoch k. The projection feeds the atlas and the next brief, so every lane can see what the others have already tried, with the pseudocode given in Appendix B.

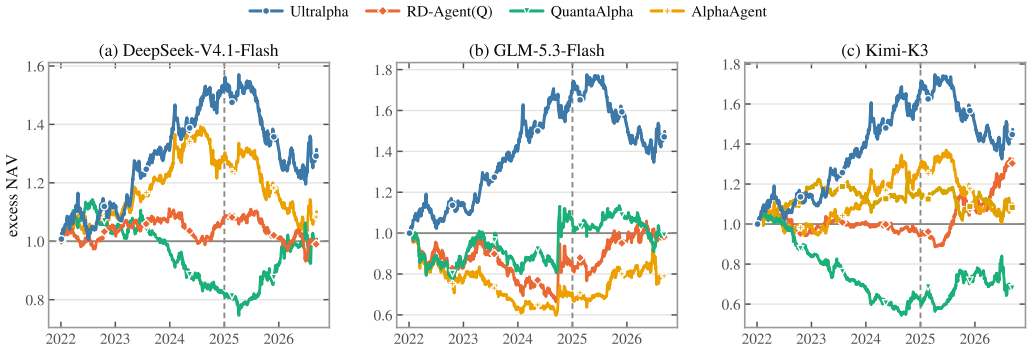

Figure 3: Wealth of each method's development top-1 factor relative to the equal-weight universe. Each panel compares the ULTRALPHA factor with the native baselines on the same backbone. ULTRALPHA earns +6.2% to +9.2% ARR with 16% to 18% maximum drawdowns.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Market and tasks. We select the A-share CSI300 universe as the evaluation market. The A-share adapter serves point-in-time prices, volume, and adjustment factors for the CSI300 universe from a frozen community data source, with prediction horizons of 24h, 120h, and 480h. The splits use 2008 through 2021 for D_{train}, 2022 through 2024 for D_{dev}, and January 1, 2025 through September 11, 2026 for D_{test}. On these windows we compare the delivered factors, the single factors each method mines, in diversity and in predictive quality. A factor is delivered only when the run returns it and the evaluator can score it. Failed attempts are left out, and identical code is counted once.

Baselines. The baseline methods are RD-AGENT(Q)-FACTOR (Li et al., 2025), an agentic research and development loop, QUANTAALPHA (Han et al., 2026), an evolutionary search over mining trajectories, ALPHAAGENT (Tang et al., 2025), a regularized factor-library agent, ALPHAFORGE-AFF (Shi et al., 2025), a population-based symbolic generator, and VANILLA GP, a plain genetic programming search. The first three are backbone-matched agentic policies. ALPHAFORGE-AFF and VANILLA GP are model-free symbolic controls. Native runs retain their search procedures, while common readouts evaluate their delivered factors, with the ULTRALPHA rows taken as four-epoch snapshots of one shared run and the baselines run under the parameters of their original papers and a 24-hour mining cap on the matching backbone.

Evaluation setup. Rank IC is the Spearman correlation between factor scores and forward returns (Kakushadze, 2016). Dev IC and Test IC average up to ten delivered factors with the highest development Rank IC, and a run with fewer than ten uses its full set. Table 3 reports that average at each horizon. Each factor is oriented by its development sign before averaging, and Test IC is that average on D_{test}. To compare the diversity and the surviving signal of the factors each method mines, we follow the analysis in Section 2.2 and report two counts. Kept is the number of delivered factors that remain after deduplication at $|\rho| = 0.9$ on D_{dev} and still have positive 24h test IC under that sign. Retention divides Kept by the number of delivered factors. To compare the returns of the mined factors, we report ARR (%), MDD (%)↓, and Sharpe of each method's development top-1 factor, ranked first on development data before any test measurement, over 2022-01 through 2026-09-11. ARR annualizes the growth of portfolio wealth. MDD is the largest decline from a previous high over that window. Sharpe divides the mean daily return by its standard deviation and multiplies by $\sqrt{252}$. Figure 3 plots this top-1 factor's wealth relative to the equal-weight universe, one panel per backbone.

4.2 MAIN RESULTS

Table 1 reports the frozen evaluation of the delivered factors on the A-share CSI300 universe. Figure 3 plots the top-1 factors. Three observations follow.

- ❶ **ULTRALPHA turns sustained search into surviving factors.** Comparing the retention column across methods, we can see that ULTRALPHA mines more diverse and more novel factors than the baselines. Retention reaches 95.8%, 97.0%, and 95.7% across the three backbones, while no agentic baseline exceeds 95.2%. This comes from two designs of the harness: the ledger

Figure 4: Width scaling with DEEPSEEK-V4.1-FLASH on A-share CSI 300 over four epochs. Delivered, Kept, and Ret. follow Table 1, and the IC columns are the Top-10 24h averages. The best value in each column is in **bold**. Kept grows from 9 to 80 with width.

W	Deliv.	Kept	Ret. (%)	Dev IC	Test IC
1	9	9	**100**	0.0245	+0.0259
2	25	22	88.0	0.0569	+0.0277
4	48	46	95.8	0.0576	+0.0289
8	**91**	**80**	87.9	**0.0659**	**+0.0316**

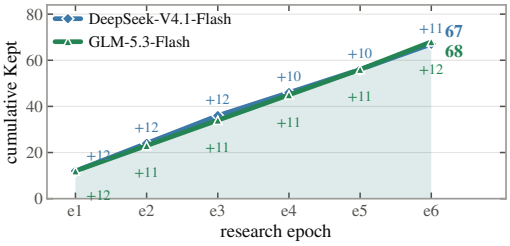

Figure 5: Depth scaling at $W{=}4$. Cumulative Kept factors over six epochs of the DEEPSEEK-V4.1-FLASH and GLM-5.3-FLASH runs, with per-epoch increments labeled.

remembers what the search has already found, so new proposals are not rewrites of earlier ones, and the gate meters the development evidence, so factors that only fit the window do not enter the delivered set. ALPHAFORGE-AFF retains 100%, because its train-only selection filters a large expression population down to five delivered factors. ALPHAAGENT keeps more factors than ULTRALPHA under DEEPSEEK-V4.1-FLASH and KIMI-K3, because it explores more and delivers more. However, 25.7% and 23.3% of its deliveries are rewrites or lose their positive test IC, against 3.0% to 4.3% for ULTRALPHA.

- ❷ **ULTRALPHA factors keep predicting outside the development window.** From the IC columns, we can read how much of the development strength survives. Dev IC averages 0.0536 to 0.0576 and leads every backbone block, 0.0134 above the closest baseline, and the same Top-10 average stays positive on the test window in all three blocks, from +0.0227 to +0.0406. This transfer comes from the same gate: a factor must prove its signal on metered development evidence before it is delivered, and the test window is never used in mining or selection, so the delivered factors keep ordering assets they have never seen.
- ❸ **ULTRALPHA earns the highest return at half the market's drawdown.** Comparing the return with the drawdown, we can see that the lead is earned in the hard years, not in the rally. The development top-1 factor of ULTRALPHA earns up to +9.2% ARR with an 18% drawdown, and all three backbone rows lead their blocks at +6.2% to +9.2% ARR with 16% to 18% drawdowns, while the equal-weight universe annualizes only +0.2% over the same window, after losing 17.5% in 2022 and 7.4% in 2023 and falling 31% from its high. The curves in Figure 3 show where the lead comes from. ULTRALPHA gains 11% to 23% a year over the universe from 2022 through 2024, and gives back 11% to 15% of it in the 2025 rally. The baselines rise and fade at +6.4% ARR or less with drawdowns of 19% to 62%, and ALPHAFORGE-AFF is the closest follower at +7.2% ARR and −19% MDD from a pool of five.

5 FURTHER ANALYSIS

5.1 WIDTH AND DEPTH SCALING

To ask whether more research compute keeps mining new factors, we vary the number of parallel research lanes and follow the surviving factors as runs lengthen.

Experimental setup. We rerun DEEPSEEK-V4.1-FLASH runs at $W \in \{1, 2, 4, 8\}$ lanes for four epochs, with three development looks per lane per epoch renewed each epoch. Total compute therefore grows with width, and the comparison measures output at increasing resources rather than compute efficiency. The $W{=}4$ arm runs on the official DEEPSEEK-V4.1-FLASH endpoint, which differs from the endpoint of the other arms, so its row reads as a robustness check. Width rows reuse the readout of Table 1 on the delivered factors of each arm. Depth follows this run and a parallel GLM-5.3-FLASH run over their six completed epochs. All comparisons reuse development data and do not match total compute.

Key findings. From the width rows of Table 4, we can see that more lanes mine more surviving factors, not more noise. Kept rises from 9 at $W{=}1$ to 22, 46, and 80 at $W{=}2$, $W{=}4$, and $W{=}8$, a steady yield of 9 to 12 factors per lane, while retention stays between 87.9% and 100%. The quality

of the leading factors does not fall with width. The Top-10 test IC rises from $+0.0259$ at $W=1$ to $+0.0316$ at $W=8$. From the depth curves of Figure 5, we can see that longer runs keep mining new factors without exhaustion. Both runs add 10 to 12 Kept factors in each of their six epochs, the increments do not shrink, and they reach 67 and 68 by epoch six. From the results, we find that ULTRALPHA scales with research compute and keeps mining meaningful new factors in both width and depth at every budget we test.

5.2 TRANSFER TO PERPETUAL CRYPTO MARKETS

To test whether the research process transfers beyond the A-share panel, we run it unchanged on eight perpetual swap markets with one-minute bars, with the result in Figure 6.



Figure 6: Development top-1 factor mined by ULTRALPHA and traded alone under the frozen construction. It keeps rising on the test set while all baselines fade.

Experimental setup. The panel pairs BTC, ETH, BNB, SOL, XRP, ADA, DOGE, and LTC with funding rate, open interest, and taker flow. We adopt KIMI-K3 to run 2 epochs with 1, 4, 8, and 24h horizons, a training window of 2020 through 2024, a development window of 2025, and a test segment from January 2026 through July 2026. Following Section 4, one development Top_1 factor is selected for analysis, here the delivered factor with the highest development 24h Sharpe, and it trades alone under one frozen long-short construction, rebalanced every 24 hours, filled at the next bar close, net of 6bps with real funding settlements. The baselines of Section 4 are designed for daily-frequency trading and cannot run on minute-frequency markets, so the reference curves are three hand-written rules under the same construction, 24h reversal, funding carry, and 7d momentum. Figure 6 plots net asset value over the combined window, with the test segment marked.

Key findings. The mined factor works, and it keeps working on data the search never saw. Over the combined development and test window it earns +28.6% with a 15.9% maximum drawdown, and on the unseen test segment it gains a further +8.5% while the underlying market falls. The three reference rules all lose money under the same construction, from -15.9% for 24h reversal to -33.4% for 7d momentum, and none recovers after the test begins. The closest relative of the selected factor, the 24h reversal rule, loses 19.2% in the test segment, so the mined factor is not a rewrite of the one-line rule it resembles.

5.3 MODEL BEHAVIOR ANALYSIS

Table 2 shows that the models behave differently while mining. In this section, we go deeper into their trajectories to see how each model actually works on the same harness.

Experimental setup. We compare the three backbones from three angles, all read from the verified ledger rather than from agent self-reports. Family persistence tells us whether a model stays on one theme or roams across mechanisms. Candidates per hypothesis tells us how many attempts one idea survives. The response to rejection tells us whether the model retunes the same hypothesis or switches family.

Key findings. The harness fixes the procedure of mining. All three models exhaust the development budget, none resubmits identical code, and about 97% of candidates are written in the DSL, so the remaining differences are in how each model works. We can see this from the family labels. GLM-5.3-FLASH is the most persistent, keeping 79.9% of consecutive attributed candidates in one family and returning the same hypothesis after 53% of rejections. DEEPSEEK-V4.1-FLASH roams instead, with 1.63 candidates per hypothesis. KIMI-K3 focuses progressively on a quiet low-volatility theme that supplies 5 of its 32 kept candidates. These findings show that the Ultralpha framework can mine valuable factors whatever the backbone's mining style, because the procedure belongs to the harness rather than to any one model.

Table 2: Behavior and resource use over four A-share CSI300 epochs with the same development allowance. Tokens per kept factor differ $2.4\times$, while kept factors per active hour differ $1.6\times$.

Model	Kept	Same-family	Cand./hyp.	Train-clean	Dev hit	Analysis queries	Tokens/Kept (M)	Kept/hour
KIMI-K3	32	0.705	2.84	**58.7%**	**47.9%**	**214**	**10.4**	2.32
DEEPSEEK-V4.1-FLASH	**46**	0.635	1.63	45.6%	31.2%	114	25.2	**3.71**
GLM-5.3-FLASH	45	**0.799**	2.47	52.8%	37.5%	73	16.6	3.55

6 RELATED WORK

Agentic quantitative research. Early alpha mining methods search over formula trees with reinforcement learning or population-based operators (Yu et al., 2023; Shi et al., 2025; Chen et al., 2026). More recent systems use language models to write and refine factor programs. ALPHA-GPT supports interactive human and model mining (Wang et al., 2025), while QUANTAGENT runs a self-improving search loop (Wang et al., 2024). Subsequent work expands the search procedure in different directions. ALPHAAGENT regularizes exploration against alpha decay (Tang et al., 2025), RD-AGENT(Q) jointly develops factors and predictive models (Li et al., 2025), and QUANTAALPHA evolves complete mining trajectories (Han et al., 2026). Other methods add agentic reinforcement learning (Tang et al., 2026), multi-agent construction (Liu et al., 2026b; Weng et al., 2026), or research memory across mining cycles (Liu et al., 2026a). ALPHABENCH further separates alpha mining into distinct task stages (Luo et al., 2026). Where prior alpha systems focus on factor generation or trajectory search, ULTRALPHA treats the research process over longer horizons by carrying verified state across epochs, organizing mechanism families with an opportunity atlas, and metering development feedback inside a host-controlled loop.

Autonomous research agents and adaptive evaluation. Outside quantitative finance, agents search over code and scientific workflows. AIDE explores solution programs (Jiang et al., 2025), MLE-BENCH and MLGYM evaluate machine-learning engineering agents (Chan et al., 2025; Nathani et al., 2025), and AGENT LABORATORY and the AI SCIENTIST automate parts of the research pipeline (Schmidgall et al., 2025; Lu et al., 2024; Yamada et al., 2025). ALPHAEVOLVE applies coding agents to algorithmic and scientific discovery (Novikov et al., 2025), while JIT-AGENT treats the agent harness itself as an object of optimization (Zhang et al., 2026). These settings usually score a submitted solution with a task evaluator. Financial factor mining has a different evaluation problem because feedback is derived from future returns and can change which candidates the search selects. Work on data snooping, backtest selection, and factor discovery documents this risk (White, 2000; Bailey & de Prado, 2014; Harvey et al., 2016; McLean & Pontiff, 2016; Gençay, 2026). Adaptive-data analyses further show that repeated queries can erode the validity of a finite holdout (Dwork et al., 2015; Blum & Hardt, 2015). ULTRALPHA brings this concern into the runtime by separating training and development feedback from a locked judge, binding records to candidate identity and lineage, and limiting the feedback exposed during research.

7 CONCLUSION

In this paper, we ask what sustained alpha mining becomes when it never stops. We propose ULTRALPHA, a closed-loop harness that keeps an LLM agent mining in one protected loop, with a ledger that remembers what the search has found and a gate that meters development evidence. On the A-share CSI300 universe with three base models, after deduplication 32 to 46 of its delivered factors stay positive on the test window, a retention of 95.7% to 97.0% of all delivered factors, and its Top-10 average 24h development IC of 0.0536 to 0.0576 stays positive on the test window in all three blocks. Its development top-1 factors earn 6.2% to 9.2% annualized returns at half the market's drawdown, and the same loop transfers to minute-level crypto markets and keeps mining as research compute scales. A statistically certified claim still requires a frozen candidate set and an independent test evaluation. We hope this work lays a solid foundation for future unbounded alpha mining.

AI USE STATEMENT

OpenAI Codex was used to provide feedback on experimental methodology and design, and to assist with software implementation, table and figure formatting, and manuscript drafting and editing. The authors reviewed and verified all AI-assisted code, analyses, figures, tables, and text and take responsibility for the final content.

REFERENCES

David H. Bailey and Marcos López de Prado. The deflated Sharpe ratio: Correcting for selection bias, backtest overfitting, and non-normality. *The Journal of Portfolio Management*, 40(5):94–107, 2014.

David H. Bailey, Jonathan M. Borwein, Marcos López de Prado, and Qiji Jim Zhu. The probability of backtest overfitting. *The Journal of Computational Finance*, 20(4):39–69, 2017. doi: 10.21314/JCF.2016.322.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, 57(1):289–300, 1995.

Avrim Blum and Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning*, 2015.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025.

Binqi Chen, Hongjun Ding, Ning Shen, Taian Guo, Jinsheng Huang, Luchen Liu, and Ming Zhang. AlphaSAGE: Structure-aware alpha mining via GFlowNets for robust exploration. In *International Conference on Learning Representations*, 2026.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

Eray Gençay. What survives honest evaluation? leakage-safe, search-aware assessment of LLM-driven trading strategy discovery. *arXiv preprint arXiv:2608.27734*, 2026.

Jun Han, Shuo Zhang, Wei Li, Yifan Dong, Tu Hu, Yumo Zhu, Xiaomin Yu, Xin Guo, Zhaowei Liu, Kunyi Wang, Jingping Liu, Tianyi Jiang, Ruichuan An, Sen Hu, Zhi Yang, Ronghao Che, and Huacan Wang. QuantaAlpha: An evolutionary framework for LLM-driven alpha mining, 2026. URL `https://arxiv.org/abs/2602.07085`.

Campbell R. Harvey, Yan Liu, and Heqing Zhu. ... and the cross-section of expected returns. *The Review of Financial Studies*, 29(1):5–68, 2016.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016.

Yuante Li, Xu Yang, Xiao Yang, Minrui Xu, Xisen Wang, Weiqing Liu, and Jiang Bian. R&D-Agent-Quant: A multi-agent framework for data-centric factors and model joint optimization. In *Advances in Neural Information Processing Systems*, 2025.

Fengyuan Liu, Yuchen Fu, Yuqi Wang, and Qi Liu. XAlpha: A memory-driven AI quant researcher for hypothesis-to-code alpha discovery. *arXiv preprint arXiv:2607.08332*, 2026a.

Fengyuan Liu, Yi Huang, Sichun Luo, Yuqi Wang, Yazheng Yang, Xinye Li, Zefa Hu, Junlan Feng, and Qi Liu. Cognitive alpha mining via LLM-driven code-based evolution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026b. URL https://arxiv.org/abs/2511.18850.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Haochen Luo, Ho Tin Ko, Jiandong Chen, David Sun, Yuan Zhang, and Chen Liu. AlphaBench: Benchmarking large language models in formulaic alpha factor mining. In *International Conference on Learning Representations*, 2026.

R. David McLean and Jeffrey Pontiff. Does academic research destroy stock return predictability? *The Journal of Finance*, 71(1):5–32, 2016.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, et al. MLGym: A new framework and benchmark for advancing AI research agents. In *Second Conference on Language Modeling*, 2025.

Alexander Novikov et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, et al. Agent laboratory: Using LLM agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

Hao Shi, Weili Song, Xinting Zhang, Jiahe Shi, Cuicui Luo, Xiang Ao, Hamid Arian, and Luis Angel Seco. AlphaForge: A framework to mine and dynamically combine formulaic alpha factors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin. AlphaAgent: LLM-driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

Ziyi Tang, Xuexiong Yin, Weixing Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. AlphaAgentEvo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *International Conference on Learning Representations*, 2026.

Peter J. Tanous. *Investment Gurus: A Road Map to Wealth from the World's Best Money Managers*. New York Institute of Finance, 1997.

Saizhuo Wang, Hang Yuan, Lionel M. Ni, and Jian Guo. QuantAgent: Seeking holy grail in trading by self-improving large language model. *arXiv preprint arXiv:2402.03755*, 2024.

Saizhuo Wang, Hang Yuan, Leon Zhou, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Alpha-GPT: Human-AI interactive alpha mining for quantitative investment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 196–206. Association for Computational Linguistics, 2025.

Zhangyuhua Weng, Shengli Zhang, Taotao Wang, and Yihan Xia. AlphaLogics: A market logic-driven multi-agent system for scalable and interpretable alpha factor generation. *arXiv preprint arXiv:2603.20247*, 2026.

Halbert White. A reality check for data snooping. *Econometrica*, 68(5):1097–1126, 2000.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. Qlib: An AI-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*, 2020.

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

Guibin Zhang, Leo Lu, Fangzhou Xie, Kang Zhu, Junhao Wang, Zhifei Xie, Zhaochen Yu, Zihang Liu, Zhongxiang Sun, Qiankun Li, Yue Liao, Heng Chang, Xiaobin Hu, Qibing Ren, Wangchunshu Zhou, Chuanrui Hu, Yafeng Deng, and Shuicheng Yan. JIT-Agent: Scaling harness intelligence via just-in-time harness evolution. *arXiv preprint arXiv:2608.25593*, 2026.

A LIMITATIONS

Two boundaries of the current framework remain open. First, ULTRALPHA already carries verified research memory across sessions and mines without an external stopping rule, but our runs span days rather than weeks. Running the harness continuously for a week or longer, and observing whether sustained search can cover the space of hypotheses available to a human researcher, is a natural next experiment we have not yet performed. Second, the framework mines single factors. Composing factors into portfolios is equally interesting and fits the loop, since a composite is itself a candidate that the host can register, screen, and certify. The current design does not cover composition, and future work will bring it into the same certified loop.

B THE PSEUDOCODE OF OUR PROPOSED ULTRALPHA

Algorithm 1 The ULTRALPHA research loop. A run uses W lanes per epoch for K epochs. The model sees only the inner loop and its brief, and every budgeted evaluation is a host operation.

Require: task τ, backbone π, lanes W, epochs K, per-epoch development allowance
Ensure: frozen candidate set, verified research state, opportunity atlas

$V_0 \leftarrow \emptyset$, $A_0 \leftarrow \emptyset$
for epoch $k = 1$ to K **do**
  for lane $w = 1$ to W **in parallel do**
    open a fresh model thread with brief $b_k = B(V_{k-1}, A_{k-1})$
    repeat
      the model chooses $a_r \in \{$explore, hypothesize, implement, screen, develop, revise, abstain$\}$
      the host executes a_r and appends the record to the lane ledger $L^{(w)}$
      if $a_r = \text{SCREEN}(f)$ **then**
        evaluate f on the training window in one causal pass
        return outcome signs, failure classes, and cell-level reasons ▷ free
      else if $a_r = \text{DEVELOP}(f)$ **and** the allowance holds **then**
        run f on the strict causal path and charge one development look
        return gate checks on execution, predictive strength, stability, costs, and overlap ▷ $p_{(i)} \le i\alpha/m$
      end if
    until the shift or the budget ends
    seal the lane's run state ▷ also on interruption
    the supervisor ingests the sealed run into the shared store under one lock
  end for
  $V_k \leftarrow R\big(\bigcup_{w=1}^{W} L_{<k}^{(w)}\big)$ and hash it ▷ run-level projection
  update the atlas with novelty classes in the append-only ledger
end for
return the frozen set, V_K, and A_K

C MINER BEHAVIOR DIAGNOSTICS

This section expands the model behavior analysis of Section 5 with the full per-dimension results. Every quantity is computed from the verified ledger, the atlas novelty ledger, and the per-run host journals of the three A-share CSI 300 run roots over epochs one through four, with no re-evaluation and no agent self-reported status. Reading from these records is the same evidence rule the harness applies at runtime, where the ledger remembers what the search found and a claim counts only while the record holds no rejection. Two validity notes apply throughout. First, family labels come from host-side attribution because agent self-reports are unreliable. KIMI-K3 leaves 59% of its candidates without a self-reported family, improving from 88.8% in epoch one to 18.1% in epoch four, against 3% for the other two models. Second, all gate-accepted candidates in this batch are diagnostic-only and none is submittable under the batch profile, so gate acceptance denotes in-gate recognition rather than deployability, identically for all three models.

Table 3: Mean Rank IC of the top delivered factors on A-share CSI 300 at 24h, 120h, and 480h. The best value per column in each block is in bold. ULTRALPHA attains the highest development average at all three horizons and, in the GLM-5.3-FLASH block, the highest test average as well.

		Dev IC			Test IC		
Backbone	**Method**	**24h**	**120h**	**480h**	**24h**	**120h**	**480h**
DEEPSEEK-V4.1-FLASH							
	ULTRALPHA	**0.0576**	**0.0813**	**0.1314**	**+0.0289**	**+0.0210**	**+0.0097**
	RD-Agent	0.0231	0.0275	0.0427	+0.0175	+0.0036	+0.0018
	AlphaAgent	0.0300	0.0301	0.0459	+0.0244	+0.0135	+0.0080
	QuantaAlpha	0.0140	0.0236	0.0376	+0.0141	+0.0027	−0.0058
KIMI-K3							
	ULTRALPHA	**0.0567**	**0.0799**	**0.1275**	+0.0227	**+0.0090**	−0.0047
	RD-Agent	0.0185	0.0198	0.0240	+0.0133	−0.0053	−0.0031
	AlphaAgent	0.0277	0.0321	0.0497	**+0.0243**	+0.0042	−0.0050
	QuantaAlpha	0.0148	0.0230	0.0282	+0.0110	+0.0088	**+0.0104**
GLM-5.3-FLASH							
	ULTRALPHA	**0.0536**	**0.0678**	**0.1186**	**+0.0406**	**+0.0242**	**+0.0171**
	RD-Agent	0.0148	0.0190	0.0299	+0.0085	+0.0026	+0.0084
	AlphaAgent	0.0158	0.0187	0.0240	+0.0237	−0.0041	−0.0038
	QuantaAlpha	0.0402	0.0503	0.0791	+0.0130	+0.0100	+0.0046
Model-free (backbone-agnostic)							
	AlphaForge-AFF	0.0246	**0.0473**	**0.0710**	+0.0164	**+0.0108**	**+0.0110**
	Vanilla GP	**0.0334**	0.0313	0.0487	**+0.0360**	+0.0037	−0.0078

C.1 STRATEGY AND TOOL USE

To understand how each model works once the harness fixes the procedure, we read the strategy and tool-use records of the three runs. The development budget is binding for all three models because the gate meters every look, so strategy differences appear in how candidates are generated, not in how many looks are spent. KIMI-K3 issues 214 host analysis queries such as placebo tests, event studies, and lead-lag decay, against 114 for DEEPSEEK-V4.1-FLASH and 73 for GLM-5.3-FLASH, and it iterates 2.84 candidates per hypothesis against DEEPSEEK-V4.1-FLASH's 1.63. The response to rejection separates them further. After a train rejection, GLM-5.3-FLASH retunes the same hypothesis 53% of the time and DEEPSEEK-V4.1-FLASH switches family 33% of the time, while after a failed development look DEEPSEEK-V4.1-FLASH returns to duplicate-class candidates 20.6% of the time against 8.0% for KIMI-K3. No model ever resubmits identical code, and the host replay path is never triggered across 2,478 screens. Style fingerprints from the registered programs are equally distinct. KIMI-K3 uses standard-deviation and correlation operators in 36% and 17% of candidates, the highest of the three, and favors equal-weight cross-sectional composites. DEEPSEEK-V4.1-FLASH uses log, delta, and cross-sectional demeaning in 27%, 32%, and 24% of candidates. GLM-5.3-FLASH is the plainest, applying a cross-sectional rank in 92% of candidates with a five-bar window in 38.6% of windowed calls. All runs write research notes and most cite the host state as authoritative, with explicit research-state references in 16 of 18 KIMI-K3 runs, 16 of 16 DEEPSEEK-V4.1-FLASH runs, and 13 of 17 GLM-5.3-FLASH runs, while cross-shift note inheritance is strongest for KIMI-K3 at 50% of runs against 24% for GLM-5.3-FLASH. The notes matter for the loop rather than for presentation, because the research state of Section 3.2 carries memory across epochs only when the models consult it, and these counts show that they do.

D PROMPTS AND MINING TRAJECTORIES

This section documents the agent-facing interface for the A-share CSI300 runs: the task book, epoch messages, heartbeat templates, and a development-feedback excerpt. The task book is divided into four boxes for readability; the instructions retain the wording supplied in the manuscript. Explanatory text outside the boxes clarifies their scope and is not an additional runtime prompt. Symbols such as i, N, H, and the family-list placeholders denote values supplied at runtime. Appendix D.4 separately presents the three lane trajectories as examples of observed behavior.

Instructions, enforcement, and evidence. The model reads instructions and proposes research actions. The host executes those actions, meters development access, and returns bounded feedback.

Instructions to avoid lookahead or respect a budget describe the interface; enforcement comes from the corresponding host checks and ledger. Model-authored notes record leads and interpretations, while host-recorded outcomes determine candidate status. The prompt text and the observed trajectories serve different purposes: the former specifies what the agent is asked to do, and the latter shows what it did.

D.1 The Task Book

The task book is the fixed workspace contract for each lane. It specifies the data view, available actions, candidate interfaces, and research discipline. It is read together with the epoch message and inherited research material, rather than serving as the sole source of context.

Budget terminology. In the task book, a "free" training screen means that it does not spend a development look; computation, model calls, and wall-clock time still incur costs. The three-look allowance is per lane per epoch in the reported setting. For $W = 4$, this permits at most twelve looks across the lanes in one epoch. Renewing the allowance permits further adaptive use of the same development window; it does not create a fresh holdout. The host-reported balance determines which requests remain available.

Task book: data and evaluation scope

Data. Dataset `csi300/cn-daily-qlib-v1`, daily bars, with the `close` field and label horizons of 24, 120, and 480 hours. You may only use these windows. `train_explore` runs 2008-01-01 to 2021-12-31 and `dev_confirm` runs 2022-01-01 to 2024-12-31.

Gate-eligible cells (generated from the calibration manifest). 24h and 120h have calibration coverage and can produce real gate verdicts. 480h has coverage but is structurally undersampled, with at most about 54 independent dev observations against the floor of 100, so it cannot pass and serves diagnostics only.

Fields. `open`, `high`, `low`, `close`, `vwap` (raw prices), `volume`, `amount`, `factor` (adjustment feature). Field names are accepted exactly as listed.

Task book: actions and evidence

Tool surface. The workspace `bin/ultralpha-tool` is the only research tool, called as `bin/ultralpha-tool <action> '<json-args>'`. Responses are JSON with status, observation, and error. Feedback is coarse, and exact metrics are never given. Heavy analysis can take minutes, so the client waits 900 seconds by default and accepts `--timeout <seconds>`.

Action catalog. `census_anomaly_map`; `event_study`; `lead_lag_decay`; `placebo_test`; `atom_vs_combo_ablation`; `residual_incremental_test`; `parameter_stability`; `research_state` (read-only); `register_hypothesis` (observation, actor or constraint, mechanism, competing explanation, direction, primary horizon, proxy, falsifier, and cost failure condition are required); `register_candidate` and `register_dsl_candidate`; `screen_candidate`; `zoo_precheck`; `run_factor_lab`; `submit_candidate`; `falsify`; `abstain`; `stop`. Signal and outcome arguments accept raw field names only, not DSL expressions or candidate ids.

Research flow. `register_hypothesis` → `register_candidate` → `screen_candidate` (train screen, free, exact metrics never leave the host) → `run_factor_lab` (development confirmation, 3 per epoch, only after a screen pass) → `submit_candidate` / `falsify` / `abstain` → `stop`.

Evidence chain. Successful evaluation actions return an `evidence_id` that the host attaches to the evidence DAG. Terminal actions must cite these ids as `parents`. A falsify needs at least one falsifier evidence from a bound `placebo_test`, an abstain needs at least one substantive evidence, and the `terminal_evidence_id` must be new.

Task book: candidate and code interfaces

Candidate formats. Default to a DSL expression through `register_dsl_candidate`, which compiles to a causal vectorized program and screens in milliseconds. Use `register_candidate` with dual entries, `generate_factors(panel)` for the fast screen path and `initialize/update` for the causal production path, when the logic needs custom state. The two entries must be semantically equivalent.

Code contract. Read fields only through the `field()` accessor, which exposes the current cross-section. Direct indexing of the data object is rejected by static analysis. Windows come from `row.window(field, N)`, which returns the trailing frame ending at t and never beyond. Missing values are allowed only as a leading warmup prefix. Lookahead constructions, negative shifts, centered rolling, full-sample normalization, and backfilling are rejected.

DSL grammar. Constructor calls only, such as `Sub(a, b)` and `Mean(x, N)`, with no arbitrary Python and no infix operators. Whitelisted elementwise, trailing-window, pair, and cross-section operators cover arithmetic, statistics, and ranking. Expressions are parsed and compiled at registration, so causality holds by construction.

Task book: execution discipline and budget

Discipline. Use only the tool-provided data. Do not access the network, absolute paths, or exact metric values. The `.agents/` directory is host-owned. Frozen paths (`data/`, `bin/`, `.agents/`, TASK.md, `data_manifest.json`) are byte-verified at epoch closeout, and one extra file voids the shift. The sandbox rejects delete commands at policy level, so move files to `trash/` instead. About one percent of shell commands fail transiently in the sandbox layer and succeed on an immediate identical retry. Register a few high-quality candidates rather than many blind ones.

Common rejections. A refused development look (`budget_refused`) means the epoch allowance is spent, so move to a new hypothesis or train exploration instead of retrying. A static candidate rejection returns diagnostic codes, so fix field names, windows, causality, and missing data point by point. A tool timeout calls for a larger `--timeout`, not a resend. An unknown candidate or field name should be looked up, not guessed.

Development looks. Each epoch holds 3 development looks, renewed per epoch and not carried over. Spend them on the candidate most likely to pass the gate, one that stands on full train evidence, has a plausible mechanism, and targets a gate-eligible cell. Abstaining is also correct when no candidate qualifies. Every `run_factor_lab` response carries the `dev_look_balance`, and once the global remaining count reaches zero the epoch's development loop is closed.

Closeout. The epoch ends on the host's clock, not yours, so do not stop early. Finish one direction, then open the next, from census to hypothesis to candidate to confirmation. When ideas run out, return to the census and event studies for new anomalies. The workspace is the state, and notes/ is where you keep it.

D.2 Runtime Prompts

The opening message establishes the epoch, while the continuation message resumes research after a conversation segment ends. A continuation stays within the same epoch and its existing development allowance; it is not a new epoch or an allowance reset.

State at handoff. The distinction between notes and host state in the continuation prompt is deliberate. Structured research memory preserves model-authored lessons as leads. Verified research state reports outcomes reconstructed by the host, and the opportunity atlas records prior search coverage. An inherited note can suggest the next hypothesis but cannot establish that a candidate passed. The agent queries `research_state` for authoritative status and the remaining development allowance.

Epoch opening and continuation prompts

Epoch opening. This is epoch i of N for this research run. The workspace TASK.md is the complete task book, and notes/ holds your earlier research notes. Continue the research. First use the analysis tools to understand the data, then register hypotheses and candidates, and once a candidate is mature use `run_factor_lab` for development confirmation. Your allowance renews each epoch. The host ends the session when time is up, so write progress into notes/ as you go.

Continuation. This epoch still has about H hours left, timed by the host rather than by you. The end of a conversation segment is not the end of the epoch. Keep researching. If the previous direction is complete, open a new direction with a new hypothesis and new candidates. Remember that notes/ and your memory are unverified leads only. The research state returned by host tools, including screen and dev responses with their requested, canonical, and dedup fields, is authoritative, so never declare a candidate passed from memory. You can query the read-only `research_state` action for the host-recorded status and `dev_look_balance`, or re-screen before acting. Do not call `stop` early unless the host budget or time is truly exhausted.

Heartbeat messages. The host injects consolidation, redirection, and deadline messages with hashed identities. The templates below retain the manuscript's recorded wording. Consolidation requests reusable lessons; redirection requests a pivot or justification; the deadline notice requests a completed evaluation and its associated reflection. These are instructions to the agent, not evaluation receipts or evidence that an action occurred.

Heartbeat injection messages

Consolidation. HEARTBEAT CONSOLIDATION (heartbeat_consolidation_N): E experiments (or Hh wall-clock) have passed since the last consolidation. Pause new experiments briefly and consolidate using the available run-local interface. Complete every pending alpha-reflect, state the reusable lesson and the most useful next idea in its reflection, then resume exploration. The Host will ingest those reflections only after the run is sealed.

Redirection. HEARTBEAT REDIRECTION (heartbeat_redirection_N): The last K experiments produced no real improvement — the research store gained no new alive factor and the IC magnitude bucket never stepped up. The current direction appears saturated.
Coverage shows unexplored families: [*family list*].
Quarantined families (repeatedly rejected, do not resurrect without genuinely new evidence): [*family list*].
Pivot to one of them, or record a written justification for staying.

Deadline notice. HEARTBEAT DEADLINE NOTICE (heartbeat_deadline_notice_N): more than half of epoch k's Hh wall-clock has elapsed. The host rule is unchanged and machine-verified at the epoch boundary. This epoch must contain a new host-verified full Factor Lab closed by a reflection bound to that experiment. No further guidance is given — how you allocate the remaining time and budget is entirely your decision.

D.3 Feedback the Model Receives

The box shows selected fields from the host response to the first development look in the DeepSeek trajectory, for candidate `C_REV_5D_LIQ_001`. It is an excerpt, not the complete response: the budget-balance field and the full set of horizon cells are not reproduced here. The displayed fields include the verdict, submission status, coarse failure reasons, and structural eligibility flags, without exact IC values or p-values. Remaining allowance is queried through the host interface described above.

Interpreting the response. `diagnostic_complete` indicates that an evaluation completed, not that the candidate passed. `passes_gate` and `submittable` are separate fields; the `diagnostic_work_only` restriction blocks submission under this profile. The 480h structural

flag identifies an ineligible cell, rather than demonstrating that the factor has no predictive value at that horizon. The reported count of FDR-significant cells is an output of this evaluation, not a guarantee about false discoveries across the entire adaptive research run.

Development feedback for one look


```
{
"candidate_id": "C_REV_5D_LIQ_001",
"hypothesis_id": "H_REV_5D_001",
"verdict": "diagnostic_complete",
"passes_gate": false,
"submittable": false,
"submit_blocked_reasons": ["diagnostic_work_only"],
"fdr_significant_cells": 0,
"coarse_reasons": ["ic_not_significant",
"insufficient_effective_sample",
"monthly_ic_consistency_below_min",
"monthly_ic_t_mean_q10_below_min",
"spearman_ic_below_min"],
"cells": {
"C_REV_5D_LIQ_001:ret_fwd_24h": {
"status": "failing",
"reasons": ["ic_not_significant",
"monthly_ic_consistency_below_min",
"spearman_ic_below_min"]},
"C_REV_5D_LIQ_001:ret_fwd_480h": {
"status": "failing",
"structural": ["ic_metric_not_gate_eligible",
"insufficient_effective_sample"]}},
"coarse_text": "evaluation: diagnostic_complete
factor C_REV_5D_LIQ_001: coverage=low
note: exact metrics are host-only and never reported here."
}
```


D.4 Mining Trajectories of Three Backbones

To ground the model behavior analysis of Section 5 in what the agents actually did, we quote the research logs of lane `explore_g1` in epoch one of each fresh run, one trajectory per backbone.

Trajectory 1: DeepSeek-V4.1-Flash, lane `explore_g1`, epoch 1

Research queue. 1. Price reversal and volume-confirmation variants. 2. Range/volatility and lottery-payoff suppression. 3. Intraday location and overnight-gap reversal. 4. Residual incremental tests against the strongest atom.

Train screen, coarse only. The coarse train screen keeps `C_REV_5D_LIQ_001` as the best all-round candidate so far, with IC positive at all horizons and a clean 24h cell. It rejects 1-day reversal, 20-day reversal, raw volume spike, 20-day low volatility, illiquidity, skewness, and eight more simple transforms.

Development look 1 of 3. The first development look goes to `C_REV_5D_LIQ_001` and returns `diagnostic_complete` with `passes_gate=false`. All three dev cells fail, with insignificant IC, weak monthly consistency, and insufficient effective sample. The log then bars near-duplicates of this family from the remaining dev looks.

Second mechanism sweep. A second sweep tests 52-week-high proximity, 12-to-1-month momentum, overnight-gap accumulation, signed-volume flow, and return-volume correlation, and every one fails significance or consistency. Close location in the daily range is clean at 24h but weaker than the liquidity-conditioned reversal.

Data-quality lead. Because prices are raw, corporate-action jumps may contaminate reversal, so the agent first tests return construction with the adjustment factor. The adjusted variants

C_REV5_ADJMUL_001 and C_REV5_ADJDIV_001 produce the same coarse profile, so adjustment does not unlock incremental signal.

Development look 2 of 3. The second development look goes to C_REV5_RANKVOL_001, a bounded rank product of reversal and volume, and the verdict is again passes_gate=false with all three dev cells failing. The short reversal family, including liquidity interactions and adjusted-price variants, is exhausted for this epoch.

Pivot. The agent pivots to market structure and cross-sectional size, testing negative beta, negative market correlation, negative residual volatility, and low nominal price.

Development look 3 of 3. The third development look goes to C_COMP3_001, an equal-weight rank composite of 5-day reversal, close-vs-VWAP reversal, and amount variation, and it also returns passes_gate=false. The dev balance is now exhausted at lane 0 and global 0, so no valid submission is possible this epoch.

Epoch conclusion. No candidate passed the dev gate in this epoch. The strongest reproducible train structure remains 5-day reversal with liquidity conditioning, but its effect did not survive the 2022-2024 confirmation window. The next epoch starts from the failed evidence DAG rather than re-running these near-duplicate variants.

Trajectory 2: GLM-5.3-FLASH, lane explore_g1, epoch 1

Candidate 1, short-horizon reversal. The agent argues that A-share CSI300 constituents have high retail participation and constrained shorting, so short-term overreaction can produce negative autocorrelation in 1-5 day cross-sectional returns. The candidate score is the negative cross-sectional rank of the trailing 5-day adjusted-close return, with 24h as the primary horizon, 120h as the secondary horizon for decay, and 480h as diagnostic only.

Registration. The agent registers hypothesis st_reversal_v1 and DSL candidate st_reversal_5d_v1. Registration is initially rejected until primary_horizon is encoded as "24h" rather than 24.

Development look 1 of 3. The first development look returns FAIL for st_reversal_5d_v1, with 24h reasons of non-significant IC, low monthly consistency, low monthly IC t-mean, and low Spearman IC, leaving 2 dev looks. The agent moves to risk and liquidity anomalies, which it judges more plausible for stable A-share CSI300 cross-sectional pricing than the failed short-term reversal.

Refinement screens. Low-volatility and low-amount factors at 20-day windows show positive coarse IC but fail the 24h train cell, and lottery and attention factors screen similar or worse. The negative 5-day average close-vs-VWAP deviation close_vwap_reversal_v1 is the first candidate with a fully clean 24h screen cell.

Development look 2 of 3. The second development look returns FAIL for close_vwap_reversal_v1, with all horizons failing, and only 1 dev look remains.

Window diversification. Momentum and skip-momentum at 20, 60, and 120 days show negative coarse IC, which indicates medium-horizon reversal rather than momentum. Reversal ranks at 10, 20, and 40 days share one clean pattern, so the agent averages them into a composite that should be more stable than any single window.

Development look 3 of 3. The third development look returns FAIL for multi_rev_10_20_40_v1. All 3 dev looks are now used, and the three dev-tested candidates are closed with diagnostic falsifications.

Next-epoch suggestion. The log advises perturbing only the best lead, for example a 60-day volume-weighted volatility variant or its combination with the low-speculation composite, and spending a new-epoch development look only if the 24h train cell becomes fully clean. No candidate was submitted because none passed dev.

Continuation check. The agent re-reads the host state with research_state(limit=100) and finds the authoritative balance at

lane_epoch_remaining=0, so no further dev look is possible this epoch. The host labels the three dev-tested candidates dev_completed, and their terminal records are diagnostic-falsified.

Trajectory 3: KIMI-K3, lane explore_g1, epoch 1

Census findings. The panel has 3,407 trading days and 862 assets, with about 24 percent of cells missing, mostly from membership turnover. The close price looks unadjusted because the adjustment factor is not constant, so returns must use close*factor. The 480h cell is structurally undersampled, with about 54 independent observations against the floor of 100.

Atom screens. A first batch of single-atom reversal, volume-ratio, low-volatility, lottery, and Amihud factors all show positive IC signs but fail the 24h gate checks, mostly on monthly IC t-mean. Single atoms sit just below the threshold, so the agent moves to an equal-weight cross-sectional rank composite.

Composite discovery. The equal-weight cs_rank composite lifts IC above the gate threshold. A second batch adds volume volatility as the strongest single atom, and the four-atom composite c_combo_rvlm leaves only the monthly IC t-mean reject at 24h.

Gate outcome. The host-verified dev gate accepts c_combo2 with 6 atoms and c_combo3 with 7 atoms. It falsifies c_vwapdev, overnight, Amihud, volume-ratio, amount-ratio, skew, return-volume correlation, downside volatility, volume trend, and 20-day momentum. The momentum test returns negative IC at every horizon, which confirms that this market is dominated by reversal.

Construction rules for the next epoch. The log fixes three construction rules. Adjusted close equals close*factor, every factor is cross-sectionally ranked, and direction is unified so a larger value means a higher expected return. Under these rules, composite c_combo4 adds 5-day reversal and range to reach 8 atoms, and c_combo5 adds up-shadow to reach 9 atoms.

Next-epoch plan. The dev budget refreshes to 3. The plan is to dev-validate c_combo4 first, since it removes the dev-falsified VWAP deviation and adds the clean atoms 5-day reversal and range, with c_combo5 as the backup. If both pass, the agent will explore new families such as EMA deviations and volatility term structure.