AEL: Agent Evolving Learning for Open-Ended Environments

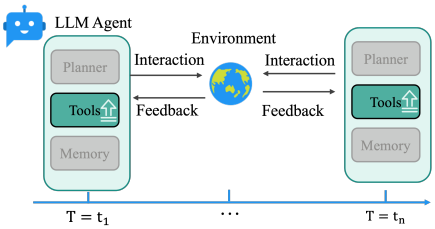
Anonymous ACL submission

Abstract

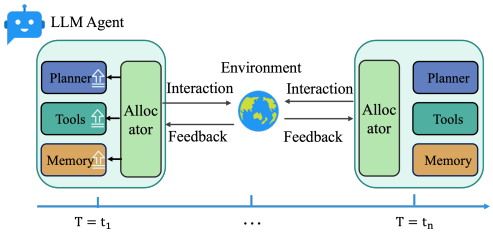
LLM agents accumulate experience but rarely learn *how to use* it: which memories to retrieve, when retrieved evidence is misleading, and when the retrieval strategy itself should change. We introduce *Agent Evolving Learning* (AEL), a two-timescale framework that recasts memory use as online policy selection. A fast Thompson Sampling bandit selects among memory-retrieval policies episode by episode, while slow LLM reflection follows a *diagnose-before-prescribe* principle: it first explains why performance degraded, then injects a targeted new retrieval policy as a bandit arm when the current pool plateaus. AEL outperforms ten self-improving and non-LLM baselines on a sequential portfolio benchmark, lifting Sharpe by 27% over the strongest memory-only variant with the lowest variance among all stochastic methods, and generalizes to a support-ticket routing stream, where it improves accuracy by 18% over reflection-free Thompson Sampling and by 51% over the best prior baseline. Mechanism studies further show that the gains are causal: reflection helps precisely when regimes demand different retrieval behavior, and is provably no-harm/no-gain when the best policy is stable. Code is available at [1].

1 Introduction

LLM agents increasingly operate in long-horizon settings such as coding (Jimenez et al., 2024), web navigation (Zhou et al., 2024), and sequential prediction (Yao et al., 2023). They typically combine planning (Huang et al., 2024), tool use (Schick et al., 2023), and memory (Xu et al., 2025), but still solve each episode with little reliable improvement from prior experience. We argue that the bottleneck is not simply storing more memory, but learning *how to use* memory: which retrieval policy to apply, when past episodes are misleading, and when the agent's strategy should change.



(a) Previous methods (Zhao et al., 2024; Yang et al., 2026).



(b) Our method.

Figure 1: Prior methods typically commit to one retrieval/update pattern. AEL instead learns *how* to retrieve experience with a fast memory-policy bandit and slow failure-diagnosis reflection.

Prior self-improving agents improve one representation at a time. Reflexion (Shinn et al., 2023) accumulates critiques, ExpeL (Zhao et al., 2024) stores lessons, Meta-Reflexion (Wu et al., 2025) distills rules, and EvoTool (Yang et al., 2026) evolves tool policies. These mechanisms are valuable, but they usually commit to a fixed retrieval or update pattern. In dynamic sequential environments, this is brittle: the memory policy that helps early may overload the planner later, while a conservative policy may miss regime-specific evidence.

We introduce *Agent Evolving Learning* (AEL), a two-timescale framework for learning memory use. At the fast timescale, Thompson Sampling (Chapelle and Li, 2011) chooses among memory-retrieval policies and updates each policy's posterior from scalar episode rewards. At the slow timescale, LLM reflection aggregates trajectories, diagnoses failure modes, and injects a causal frame into the planner prompt; when the current policy

[1] https://anonymous.4open.science/r/AEL_submission-F993

pool underperforms, reflection may add a new retrieval policy as a bandit arm. Planner and tool evolution are supported as framework extensions, but experiments show that they hurt in this short-horizon regime, so the default AEL keeps planner and tools fixed. Our contributions are as follows.

❶ ***Framework.*** We propose AEL, a two-timescale framework for learning memory use in sequential agent tasks. A fast Thompson Sampling bandit selects among memory-retrieval policies episode-by-episode, while slow LLM-driven reflection aggregates trajectories to diagnose failures and evolve the policy pool.

❷ ***Methodology.*** We introduce a *diagnose-before-prescribe* reflection mechanism: rather than accumulating experience indiscriminately or evolving modules on a fixed schedule, the LLM first builds a causal explanation of why performance degraded, then generates a targeted architectural change (e.g., a new retrieval policy injected as a bandit arm). This couples qualitative failure diagnosis with quantitative bandit selection.

❸ ***Results.*** On the D-full portfolio benchmark, AEL achieves Sharpe 2.13, outperforming ten baselines with the lowest variance among stochastic methods; on the support-ticket routing stream, AEL reaches 0.840 accuracy, outperforming the strongest prior baseline by +28.3pp.

2 Related Work

2.1 LLM Agents

ReAct (Yao et al., 2023) interleaves chain-of-thought reasoning (Wei et al., 2022) with action execution, establishing the *reason-then-act* paradigm. Tree of Thoughts (Yao et al., 2024) expands the reasoning space via branching, while Huang et al. (2024) identify planning as a key bottleneck. Toolformer (Schick et al., 2023) demonstrates autonomous API invocation. For memory, MemGPT (Packer et al., 2023) introduces OS-inspired paging between fast and slow storage, and A-Mem (Xu et al., 2025) proposes self-organized agentic memory. These three modules (planner, tools, memory) form the backbone of modern agents but are typically fixed once deployed. AEL's default configuration learns over the memory subsystem (which retrieval policy to use) while keeping the planner and tools fixed; the framework also supports contextual bandits over planners and per-tool Thompson Sampling as ablation knobs.

2.2 Evolving Learning of LLM Agents

Several methods enable agents to improve, each evolving a subset of modules. Reflexion (Shinn et al., 2023) accumulates self-critiques in the context window without structured retrieval. Voyager (Wang et al., 2023) builds a skill library through curriculum-driven exploration but does not adapt tools or planning. ExpeL (Zhao et al., 2024) extracts lessons via keyword matching. Meta-Reflexion (Wu et al., 2025) distills reflections into rules (the closest analogue to AEL's procedural memory) but neither evolves tools or planners. AutoAgent (Wang et al., 2026a) explores elastic memory with evolving cognition. EvoTool (Yang et al., 2026) evolves tool-use policies via blame-aware mutation but holds memory fixed; FactorMiner (Wang et al., 2026b) combines skills with experience memory; Tool-Genesis (Xia et al., 2026) benchmarks tool creation. None jointly evolve tools and memory with reflection-driven self-diagnosis or study credit assignment in multi-module evolution.

2.3 Bandits and Credit Assignment

Contextual bandits (Li et al., 2010; Auer et al., 2002) and Thompson Sampling (Chapelle and Li, 2011; Agrawal and Goyal, 2012) provide efficient online learning with natural exploration. For multi-party credit assignment, Shapley values (Shapley, 1953; Lundberg and Lee, 2017; Ghorbani and Zou, 2019) offer a principled cooperative-game framework but largely treat modules as black boxes. AEL systematically evaluates uniform, factored counterfactual, and LLM-driven credit assignment, finding that simpler methods often outperform in high-noise domains and further highlighting credit assignment as an important open challenge.

3 The AEL Framework

3.1 Overview and Design Rationale

Let $\{e_t\}_{t=1}^{T}$ denote the episode stream. AEL operates on two episode-based timescales: fast windows $\mathcal{W}_k^{\text{fast}}$ of size N update selection preferences, while slow windows $\mathcal{W}_j^{\text{slow}}$ of size $M \gg N$ aggregate evidence for reflection and memory consolidation. At episode t, the agent selects a configuration $c_t = (p_t, z_t, m_t)$ specifying planner, tool set, and memory retrieval policy, then observes outcome score $s_t \in [-1, 1]$. In the default AEL configuration, the planner and tools are fixed; a Thompson Sampling bandit selects among memory retrieval

Table 1: Feature comparison with prior methods. AEL is the only method combining multi-tier memory, learned retrieval, and causal diagnosis across two timescales.

Method	Memory	Retrieval Learning	Diagnosis	Dual Scale	Policy Evol.
Reflexion (Shinn et al., 2023)	Context	–	–	–	–
Voyager (Wang et al., 2023)	Skills	–	–	–	–
ExpeL (Zhao et al., 2024)	Lessons	–	–	–	–
Meta-Reflexion (Wu et al., 2025)	Rules	–	–	–	–
EvoTool (Yang et al., 2026)	–	–	blame	–	tools
FactorMiner (Wang et al., 2026b)	Experience	–	–	–	–
AEL (ours)	**3-tier memory**	✓	✓	✓	✓

policies, learning which way of accessing experience suits the current stage. At the slow timescale, LLM-driven reflection diagnoses failure patterns and injects causal insights into the decision prompt. Uniform credit converts outcomes into bandit rewards; the framework additionally supports planner selection (LinUCB), per-tool Thompson Sampling, cold-start, and skill extraction, all evaluated in Table 3. Figure 2 shows the pipeline; episodes are split chronologically into train/val/test, with all learning frozen at test time.

Components: what is learned and what is fixed. The default AEL configuration learns three quantities online: (i) a Beta posterior per memory-retrieval policy via Thompson Sampling, (ii) a reflection insight regenerated each slow window and injected into the planner prompt, and (iii) procedural rules promoted from recurring semantic patterns. The planner identity, the tool set (12 tools), and all tool implementations remain fixed. The framework additionally supports planner selection (LinUCB contextual bandit), per-tool Thompson Sampling, cold-start initialization (with uninformative priors $\alpha_0{=}\beta_0{=}1$), and skill extraction as optional extensions; we keep all of these off by default because the ablation (Table 3) shows they degrade performance in this short-horizon regime. Credit assignment uses uniform rewards; factored counterfactual (FCC) and LLM-driven (LLM-FCC) alternatives are also evaluated in Table 3 and found harmful. A reader who wants a single takeaway should read AEL as "learn how to use memory, not how to plan or what tools to call."

3.2 Memory-Policy Selection and Extended Module Bandits

The central online learning mechanism in AEL is memory-policy selection via Thompson Sampling (Chapelle and Li, 2011). Each policy m in a pool of five initial strategies maintains a Beta posterior $\mathrm{Beta}(\alpha_m, \beta_m)$; at each episode the agent samples $\tilde{\mu}_m \sim \mathrm{Beta}(\alpha_m, \beta_m)$ and selects the policy with the highest sample. After observing the episode outcome, the posterior is updated: $\alpha_m \mathrel{+}= \tilde{r}_t$, $\beta_m \mathrel{+}= (1 - \tilde{r}_t)$. The five initial policies range from no retrieval to aggressive multi-tier recall (Appendix N); new policies can be added by the evolution mechanism described in Section 3.4. This learns *how* to access experience (compressed summaries, full records, or no retrieval), adapting the strategy as the memory store matures. The framework additionally supports two further bandit levels, evaluated in the ablation (Table 3):

Planner selection (LinUCB (Li et al., 2010)). A contextual bandit whose context vector $\phi_t \in \mathbb{R}^7$ encodes sector, 30-day volatility, log market cap, data richness, momentum, options availability, and analyst coverage. It selects $\pi_t = \arg\max_\pi \big(\phi_t^\top \hat{\theta}_\pi + \alpha\sqrt{\phi_t^\top \mathbf{A}_\pi^{-1} \phi_t}\big)$, balancing exploitation of the best-performing planner with exploration of uncertain alternatives (full update equations in Appendix F).

Per-tool selection (Thompson Sampling). Each tool a maintains a Beta posterior $\mathrm{Beta}(\alpha_a, \beta_a)$ updated from directional hit/miss statistics against realized outcomes; the top-K tools by sampled value are selected, with K shrinking as learning progresses (details in Appendix F).

3.3 Three-Tier Evolving Memory

Raw episode logs are too noisy and voluminous to retrieve directly. The agent needs to progressively distill experience from specific episodes into general patterns and then into actionable rules. AEL achieves this through three memory tiers with automatic promotion.

Episodic memory records each episode's raw outcome: which tools were used, what signals they produced, whether each signal was correct, and the overall prediction quality. This provides the ground-truth training data for distillation.

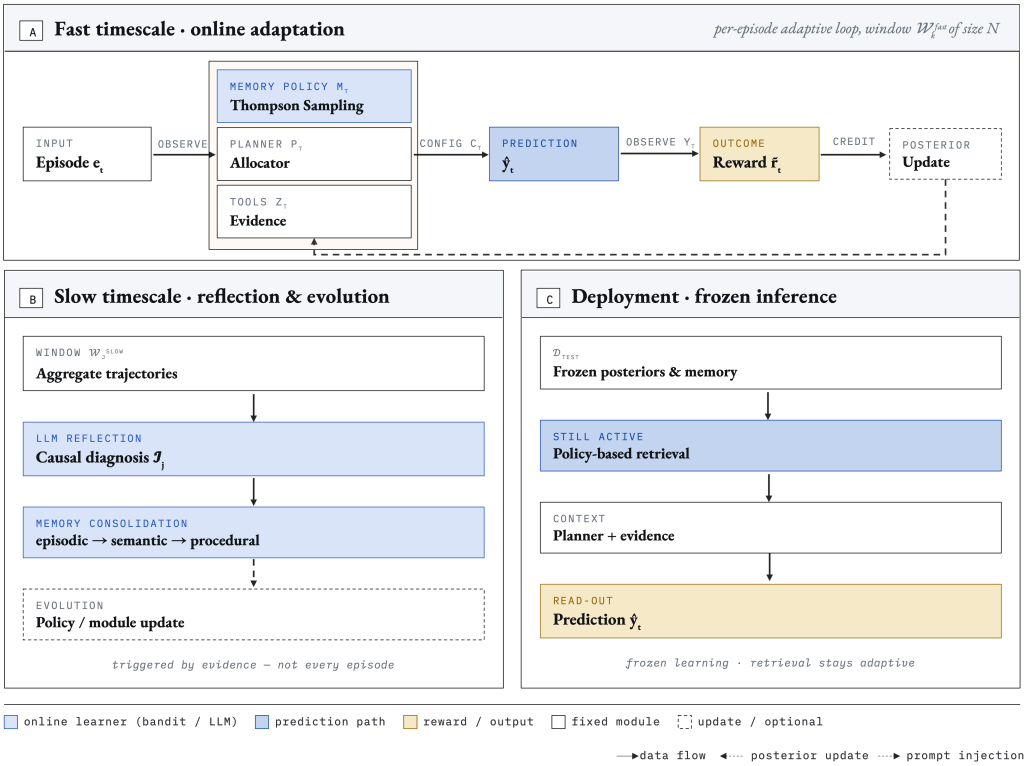


Figure 2: AEL framework overview. At the fast timescale (A), a Thompson Sampling bandit selects memory retrieval policies episode-by-episode, feeding planner-guided predictions and bandit updates. At the slow timescale (B), an LLM reflects on aggregated trajectories to consolidate memory and evolve modules; during deployment (C), all weights are frozen while policy-based retrieval remains active.

Semantic memory aggregates episodic records into cross-episode patterns, distilled periodically (every 10 episodes). These patterns capture regularities invisible in any single episode, such as "momentum indicators are reliable for trending stocks but misleading during reversals."

Procedural memory promotes high-confidence semantic patterns into executable rules injected directly into planner prompts. These rules represent the agent's most trusted knowledge, influencing behavior without requiring retrieval at query time.

Memory use involves two distinct decisions. First, the Thompson Sampling bandit selects a *retrieval policy* that specifies which tiers are visible, how many entries to retrieve, and how they are formatted (e.g., compressed summaries vs. full records vs. no retrieval at all). Second, *within* the selected policy, entries are ranked by a composite relevance score:

$$r(q,e) = \underbrace{f_{\text{match}}(q,e)}_{\text{ticker, sector, tool}} \times \underbrace{(0.5 + 0.5 q_e)}_{\text{quality}} \times \underbrace{(0.3 + 0.7 e^{-0.01\Delta})}_{\text{recency}} \times \underbrace{b_\tau}_{\text{tier boost}}$$

where f_{match} sums feature-match bonuses, q_e is the entry's quality score, Δ is the number of episodes since the entry was written, and $b_\tau \in \{1.0, 1.2, 1.5\}$ for episodic, semantic, and procedural tiers respectively. The top-k entries (default k=5) above a quality threshold are returned. This two-layer design means the bandit learns *how* to access experience, while the scoring function determines *which* entries surface under that access strategy.

3.4 LLM-Driven Reflection and Code Evolution

Selection within a fixed module pool will eventually plateau if the pool itself is inadequate. The reflection system addresses this by letting the LLM diagnose *why* performance has degraded and then generate new modules to address the diagnosed problem.

Cold-start initialization. Before any episodes, the LLM reads tool and planner descriptions and generates informed priors for the bandit system, reducing the exploration cost of uniform initialization.

Slow-window reflection. After each slow window, the LLM receives four inputs: (i) per-ticker episode summaries (planner used, score, directional accuracy), (ii) per-tool accuracy statistics aggregated over the window, (iii) market side information

(sector returns, volatility regime, cross-correlation) computed from cached price data but *not* given to the predictor, and (iv) the last three reflections for temporal continuity. It produces a structured output containing a *causal insight* (what market conditions caused today's outcomes and why specific signals were reliable or misleading), a regime label, and a confidence score. This insight is injected directly into the allocator's prompt at the next episode, giving the predictor an interpretive frame for its evidence without polluting the memory store with noisy LLM-generated narratives. Module selection remains with the bandit; reflection diagnoses conditions, not configurations.

Code evolution is triggered by diagnosed structural failures, not fixed schedules. When a planner's failure streak exceeds a threshold (default 3 consecutive slow windows), the LLM generates a new Python planner class tailored to the diagnosed failure mode; the new class must pass syntax validation and a smoke test before entering the pool. Memory policy evolution fires periodically (every 5 slow windows, starting after window 10) only when the average Thompson Sampling reward across all policies falls below 0.4, indicating that the entire policy pool is underperforming. The LLM then designs a new retrieval policy specifying tier visibility, retrieval depth, and formatting strategy, which is added as a new bandit arm. Note that in the main AEL configuration, planner evolution is disabled; it is active only in the full EAEL variant. The design principle is *diagnose before prescribe*: the LLM must first build an explanation of why performance degraded before generating an architectural change, making evolution targeted rather than random.

3.5 A Running Case: One Failure, One Reflection, One Policy Shift

On seed 42, a bear-regime loss causes the memory bandit to down-weight an `aggressive_learner` retrieval policy. After the slow window, reflection diagnoses that bull-regime memories are misleading under elevated volatility and later proposes `RegimeFilteredRetrieval`, a small validated retrieval policy that filters memories by regime metadata. The new arm enters Thompson Sampling with uninformative priors; within the next slow window the mean reward rises from 0.19 to 0.38. This trace illustrates the intended division of labor: the bandit consumes scalar rewards, while reflection consumes aggregated trajectories and side information to decide whether the policy pool itself is missing an arm. The full trace and generated code are in Appendix S and Section P.2.

Algorithm 1 The training process of AEL

Require: Episode stream $\{e_t\}_{t=1}^{T}$; memory-policy pool $\mathcal{M}=\{m_1,\ldots,m_K\}$; priors $\alpha_k{=}\beta_k{=}1$; slow-window size M; warm-up W; evolution threshold $\bar{r}_{\min}$
Ensure: Posteriors $\{(\alpha_k,\beta_k)\}$, memory $\mathcal{D}$, pool $\mathcal{M}$

Partition $\{e_t\}$ into $\mathcal{E}_{\text{train}}, \mathcal{E}_{\text{val}}, \mathcal{E}_{\text{test}}$ chronologically
for each slow window $\mathcal{W}_j \subset \mathcal{E}_{\text{train}}$ **do**
  for each episode $e_t \in \mathcal{W}_j$ **do**
    Sample $\tilde{\mu}_k \sim \text{Beta}(\alpha_k, \beta_k)$ for each $m_k \in \mathcal{M}$
    $m_t \leftarrow \arg\max_k \tilde{\mu}_k$ ▷ Thompson Sampling
    $\mathbf{x}_t \leftarrow \textsc{Retrieve}(\mathcal{D}, m_t, e_t)$ ▷ Policy-guided recall
    $\hat{y}_t \leftarrow \textsc{Plan}(e_t, \mathbf{x}_t, \mathcal{Z})$ ▷ Predict with tools $\mathcal{Z}$
    Observe y_t; compute $s_t \in [-1, 1]$
    $\tilde{r}_t \leftarrow \text{clip}\big((s_t{+}1)/2,\ 0,\ 1\big)$ ▷ Uniform credit
    α_{m_t} += $\tilde{r}_t$; β_{m_t} += $1{-}\tilde{r}_t$
    Write $(e_t, \hat{y}_t, y_t, s_t)$ to episodic tier of $\mathcal{D}$
  end for
  $\mathcal{I}_j \leftarrow \textsc{Reflect}(\mathcal{W}_j, \mathcal{D})$ ▷ LLM causal diagnosis
  Inject $\mathcal{I}_j$ into planner prompt for $\mathcal{W}_{j+1}$
  Distill episodic → semantic → procedural tiers of $\mathcal{D}$
  if $\bar{r}_{\mathcal{M}} < \bar{r}_{\min}$ **and** $j > j_{\min}$ **then**
    $m_{\text{new}} \leftarrow \textsc{EvolvePolicy}(\mathcal{I}_j, \mathcal{D})$; $\mathcal{M} \leftarrow \mathcal{M} \cup \{m_{\text{new}}\}$
  end if
end for
Select best posteriors on $\mathcal{E}_{\text{val}}$; freeze for $\mathcal{E}_{\text{test}}$

3.6 Learning Signal

The training signal combines two complementary mechanisms at different timescales. At the fast timescale, each episode produces a scalar outcome $s_t \in [-1, 1]$ that is converted into a **uniform reward** $\tilde{r}_t = \text{clip}((s_t + 1)/2, 0, 1)$ and used directly to update the memory-policy bandit posterior (and, in the full EAEL variant, tool and planner bandits as well). Uniform credit avoids compounding misattribution in our high-noise domain. We also evaluate factored counterfactual credit (FCC, combining structural, counterfactual, and Shapley attribution) and LLM-driven credit (LLM-FCC), but both degrade performance (Table 3; full formulations in Appendix G). At the slow timescale, reflection consumes aggregated trajectories to decide *when structural evolution is needed*, providing a qualitative diagnostic channel that complements the quantitative bandit signal.

4 Experiments

We design experiments to answer four research questions. **Q1**: Does AEL outperform prior self-improving agent methods? **Q2**: Which components contribute, and does adding complexity help or hurt? **Q3**: Does the two-timescale mechanism gen-

eralize beyond finance? **Q4**: Are the improvements statistically significant? Additional mechanism analyses (bandit alternatives, reflection faithfulness, procedural-memory dynamics) are in the appendix.

4.1 Experiments Setup

Datasets. We evaluate on two benchmarks. **(1)** *D-full portfolio*: 10 sector-diverse tickers spanning 7 GICS sectors at 1-hour resolution (208 episodes: 140 train / 40 val / 28 test). Training covers diverse market regimes (bull, bear, flat); the test set contains a bear-to-bull transition (full details in Appendix L). **(2)** *Support-ticket routing*: an LLM-generated stream with 8 resolution routes, produced via Gemini 3.1 Flash-Lite. The stream spans four regimes: explicit logs, shorthand drift, imbalanced incidents, and a late route-aware regression. The LLM-agent evaluation uses 160 stratified episodes per seed (40 per regime) across 5 seeds (details in Section 4.5).

Baselines. We compare: (i) four **non-LLM baselines** (equal-weight, momentum-weighted, min-variance, inverse-momentum); (ii) five **prior self-improving methods**: Reflexion (Shinn et al., 2023), ExpeL (Zhao et al., 2024), FactorMiner (Wang et al., 2026b), Meta-Reflexion (Wu et al., 2025), and EvoTool (Yang et al., 2026) (adaptation details in Appendix Q); (iii) **HyperAgent** (Zhang et al., 2026), a recursive self-modification framework (Section 5); and (iv) an **AEL incremental build** (Stateless → +Tools → +Memory → AEL). All methods share the same 12 tools (Appendix M), backbone LLM (Claude Haiku 4.5), and data split.

Evaluation & Protocol. We report four frozen test-phase metrics: Sharpe, Sortino, and Calmar ratios (risk-adjusted return from complementary perspectives) and maximum drawdown (MaxDD%); formal definitions are in Appendix D. The matched incremental build uses seeds 42, 123, 456; headline methods extend to 5 seeds. During test, all learning is disabled (frozen bandits, read-only memory, no evolution). More details are in Appendix E.

4.2 Main Results (Q1)

Table 2a reports frozen-test metrics on the D-full benchmark across five random seeds. We focus on the Sharpe ratio as the primary metric because it directly reflects the training signal (episode-level return normalized by volatility) and captures the risk-return tradeoff that is central to sequential portfolio allocation. Sortino and Calmar ratios provide complementary views on downside risk and worst-case drawdown. AEL achieves the highest Sharpe (2.13), Sortino (4.08), and Calmar (10.40), outperforming all 10 baselines with the lowest variance among all stochastic methods.

Comparison to prior methods. Among prior methods, EvoTool is the strongest on mean Sharpe (1.37) but with the highest variance, indicating that its evolutionary tool policy is sensitive to initialization. ExpeL (0.76) and Reflexion (−0.59) both exhibit extreme seed dependence: individual outlier seeds inflate their means (detailed in Appendix J). HyperAgent (0.72) achieves the lowest variance among prior methods, but its recursive code generation fails to escape the initial equal-weight template (Appendix K). The deterministic momentum-weighted baseline (1.44) outperforms all prior LLM methods on Sharpe, underscoring that LLM-based learning must overcome the noise it introduces to beat simple heuristics.

Why AEL improves. The incremental build isolates the source of improvement: memory enables cross-episode learning (+24%), and reflection further enhances performance (+27%) by diagnosing failure patterns and filtering memory quality. Unlike prior methods that accumulate experience indiscriminately (Reflexion) or evolve only one module (EvoTool), AEL uses reflection to provide the agent with an *interpretive frame* for its evidence, turning raw memory into actionable diagnostic insights. The low variance further suggests that this diagnostic mechanism generalizes across random initializations rather than depending on a seed.

Cross-domain validation on support-ticket routing. On the support-ticket routing benchmark (Table 2b), each episode is a support ticket for an LLM API service with four regimes (explicit logs, shorthand drift, imbalanced incidents, and route-aware regression). Using Gemini 3.1 Flash-Lite with zero errors or fallbacks, AEL reaches 0.840 accuracy, outperforming TS without reflection (0.711) and the strongest prior baseline (FactorMiner, 0.557). The result is consistent with the finance story: reflection helps when it introduces a retrieval behavior unavailable in the initial pool, particularly in the late route-aware regime where R3 accuracy reaches 0.885 versus 0.705 for the strongest prior baseline.

4.3 Component Synergy and Robustness (Q2)

Figure 3a reveals two findings. First, *memory and reflection form a synergistic pair*: Stateless (1.35) → +Memory (1.68) → AEL (**2.13**), with reflection

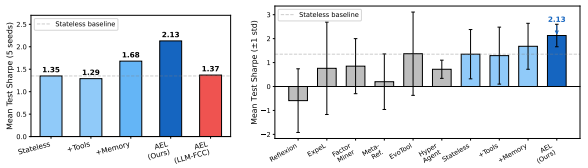
Table 2: Main results on two benchmarks (N=5 seeds for all stochastic methods). Bold = best per column.

(a) D-full portfolio benchmark. Non-LLM baselines: EqW (equal-weight), Mom (momentum-weighted), MinV (min-variance), InvM (inverse-momentum). Metric definitions in Appendix D.

Category	Method	Sharpe ↑	Sortino ↑	Calmar ↑	MaxDD% ↑
Non-LLM	EqW	0.70	1.32	3.05	−1.47
	Mom	1.44	2.73	6.89	−1.64
	MinV	−0.61	−0.82	−2.06	−1.59
	InvM	−0.20	−0.30	−0.66	−1.73
Prior	Reflexion (Shinn et al., 2023)	−0.59±1.33	−0.57±1.75	−1.64±5.92	−2.30±0.36
	ExpeL (Zhao et al., 2024)	0.76±1.93	1.68±3.93	3.97±9.79	−1.76±0.16
	FactorMiner (Wang et al., 2026b)	0.85±1.15	1.55±2.14	3.61±4.91	−1.93±0.28
	Meta-Reflection (Wu et al., 2025)	0.20±1.16	0.37±1.65	1.66±6.01	−2.22±0.23
	EvoTool (Yang et al., 2026)	1.37±1.74	2.73±3.58	6.28±7.70	**−1.46±0.15**
	HyperAgent (Zhang et al., 2026)	0.72±0.38	1.00±0.53	3.90±2.37	−2.49±0.15
AEL Variants	Stateless	1.35±1.03	2.51±1.93	6.52±4.98	−1.59±0.26
	+Tools	1.29±1.19	2.51±2.23	6.07±5.52	−1.63±0.26
	+Memory	1.68±0.96	3.29±1.83	8.50±5.09	−1.53±0.11
Ours	**AEL**	**2.13±0.47**	**4.08±1.11**	**10.40±2.75**	−1.53±0.06

(b) Support-ticket routing (Gemini 3.1 Flash-Lite, 160 stratified episodes per seed). R0–R3 are explicit logs, shorthand drift, imbalanced incidents, and route-aware regression. Baseline adaptations in Appendix V.

Category	Method	Overall	R0	R1	R2	R3
No memory	Stateless	0.512 ± 0.010	0.970	0.235	0.220	0.625
Fixed policy	Class-balanced	0.565 ± 0.036	0.965	0.290	0.465	0.540
Prior	Reflexion (Shinn et al., 2023)	0.490 ± 0.045	0.925	0.230	0.235	0.570
	ExpeL (Zhao et al., 2024)	0.539 ± 0.018	0.960	0.245	0.245	0.705
	Meta-Reflection (Wu et al., 2025)	0.497 ± 0.031	0.980	0.245	0.295	0.470
	EvoTool (Yang et al., 2026)	0.540 ± 0.046	0.980	0.370	0.435	0.375
	FactorMiner (Wang et al., 2026b)	0.557 ± 0.037	0.960	0.290	0.455	0.525
Ablation	TS no reflection	0.711 ± 0.039	**0.985**	0.590	0.585	0.685
Ours	**AEL**	**0.840 ± 0.031**	**0.985**	**0.775**	**0.715**	**0.885**



(a) Incremental component build. (b) Mean-focused robustness summary.

Figure 3: **(a) Incremental build (5 seeds):** Stateless (1.35) → +Memory (1.68) → AEL (**2.13**); adding LLM-FCC credit degrades to 1.37. **(b) Robustness:** Mean test Sharpe ±1 std. AEL has the highest mean (2.13) with the lowest variance among LLM methods.

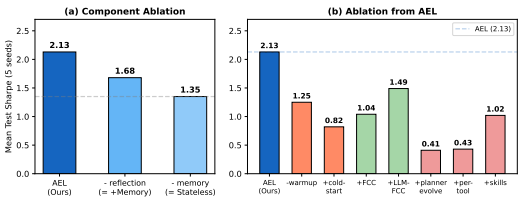


Figure 4: The ablation study.

providing a 27% jump over memory alone. The disproportionate gain from reflection confirms that the bottleneck in open-ended agent improvement is *how to use* experience, not simply accumulating it. Second, adding LLM-FCC credit to the full AEL system *degrades* performance (1.37), showing that sophisticated credit assignment introduces more noise than signal in this high-noise domain. Figure 3b confirms that AEL achieves the highest mean Sharpe with the lowest variance among LLM methods exceeding mean Sharpe 1.0, while EvoTool, +Memory, and Stateless all exhibit substantially higher seed-level variability. HyperAgent attains a lower std but at a markedly lower mean (0.72), reflecting its collapse to an equal-weight prior rather than adaptive selection.

4.4 Ablation and Credit Analysis (Q2)

Table 3 and Figure 4 reveal a striking finding: *every modification to* AEL *degrades performance*. Removing warm-up (Δ−0.88) is most damaging: without the 15-episode warm-up, the memory bandit receives noisy early rewards that corrupt Beta

Table 3: Ablation from AEL (5 seeds). Each row modifies one component. Every change degrades performance.

Configuration	Sharpe	Sortino	ΔSharpe
AEL (Ours)	**2.13**	**4.08**	—
Remove component			
− warm-up	1.25	2.25	−0.88
− reflection	1.68	3.29	−0.45
Add complexity			
+ cold-start init	0.82	1.41	−1.31
+ planner evol.	0.41	1.01	−1.72
+ per-tool sel.	0.43	0.80	−1.70
+ skill extract.	1.02	1.78	−1.11
Change credit method			
→ FCC	1.04	1.96	−1.09
→ LLM-FCC	1.49	2.84	−0.64

posteriors irreversibly, causing premature convergence to a suboptimal retrieval policy. Reflection contributes $\Delta{-}0.45$ via failure diagnosis and quality filtering; though modest in magnitude, the +Memory baseline (1.68) is already strong, so reflection's value lies in interpretive quality rather than raw gain. Adding complexity consistently hurts: planner evolution ($\Delta{-}1.72$) and per-tool selection ($\Delta{-}1.70$) suffer from data starvation (208 episodes split across 6 planners or 120 tool-ticker arms), cold-start ($\Delta{-}1.31$) introduces distributional bias, and skill extraction ($\Delta{-}1.11$) overfits to training regimes. For credit assignment, both FCC ($\Delta{-}1.09$) and LLM-FCC ($\Delta{-}0.64$) underperform uniform credit, highlighting credit assignment as an open problem in high-noise, short-horizon settings.

4.5 Cross-Domain Mechanism Transfer (Q3)

To test whether the two-timescale mechanism is domain-agnostic rather than finance-specific, we construct a deterministic, LLM-free 10-arm Bernoulli bandit with four regimes (52 episodes each, 208 total). A hidden 11th arm pays 0.70 during regime 3 but only 0.30 in regimes 0–2; it is not available at start and can only be added through reflection. Table 4 shows that AEL-style TS+reflection reaches 0.400 overall, comparable to UCB1 (0.402), but in regime 3 where the hidden arm fires, AEL reaches **0.452** versus pure TS at 0.359 — a +9.3pp lift. The mechanism transfers directionally, though the gain is smaller than in finance because the synthetic setting lacks informative regime features that reflection can exploit. A 20Newsgroups boundary case (appendix) confirms

Table 4: Synthetic cross-domain bandit (5 seeds). Reflection-driven arm injection yields +9.3pp in R3.

Algorithm	R0	R3	Overall	Sharpe
RR	.364	.352	.385±.023	.79
UCB1	.392	.410	.402±.026	.82
ε-gr	.396	.331	.364±.017	.76
TS	.372	.359	.385±.021	.79
AEL	.408	**.452**	.400±.020	**.82**
Oracle	.632	.731	.688±.011	1.49

Table 5: Welch t-test and F-test for AEL (Sharpe 2.13, n=5) vs. baselines. $^{***}p<.001$, $^{**}p<.01$, $^{*}p<.05$, $^{\bullet}p<.10$.

Baseline	mean (std)	$\Delta\mu$	t	Welch p	F p
Refl.	−0.59 (1.33)	+2.72	+4.31	.008**	.069$^{\bullet}$
ExpeL	+0.76 (1.93)	+1.37	+1.54	.190	.018*
FMiner	+0.85 (1.15)	+1.28	+2.30	.066$^{\bullet}$	.111
Meta-R	+0.20 (1.16)	+1.93	+3.45	.017*	.108
EvoTool	+1.37 (1.74)	+0.76	+0.94	.393	.027*
HyperAg	+0.72 (0.38)	+1.41	+5.22	.001***	.691
Stateless	+1.35 (1.03)	+0.78	+1.54	.178	.158
+Tools	+1.29 (1.19)	+0.84	+1.47	.200	.099$^{\bullet}$
+Memory	+1.68 (0.96)	+0.45	+0.94	.384	.195

that when the best retrieval policy is stable across regimes, the mechanism behaves as no-harm/no-gain.

4.6 Statistical Significance (Q4)

We test the headline mean-Sharpe comparison with Welch's two-sample t-test ($n_1{=}n_2{=}5$ seeds) and variance equality with an F-test on $\sigma^2_{\text{AEL}}/\sigma^2_{\text{baseline}}$. On mean Sharpe, 4 of 9 comparisons survive $\alpha{=}0.05$ (Reflexion, Meta-Reflexion, HyperAgent); the remaining baselines have large per-seed variance that inflates the pooled standard error. On variance, the F-test is significant for ExpeL ($p{=}0.018$) and EvoTool ($p{=}0.027$): the reliability difference is statistically robust even where the mean difference is not. AEL's 95% CI is $[1.72, 2.54]$; EvoTool's is $[-0.16, 2.90]$ (overlaps zero).

5 Conclusion

In short-horizon, high-noise sequential domains, learning *how to use memory* is more valuable than adding more adaptive machinery. On finance, AEL raises Sharpe to 2.13; on the support-ticket routing stream, it reaches 0.840 accuracy. The claim is structural: a fast bandit over retrieval policies plus slow reflection-driven policy-pool updates works when regimes create different memory needs, and behaves as no-harm/no-gain otherwise.

Limitation

AEL is designed for short-horizon, high-noise sequential settings, and our study is scoped accordingly: we evaluate two task families over moderate-length episode streams, and the value of reflection-driven evolution is naturally tied to the diagnostic capability of the backbone LLM. Extending the two-timescale mechanism to longer horizons, additional domains, and stronger credit-assignment signals is a promising direction for future work.

References

Shipra Agrawal and Navin Goyal. 2012. Analysis of thompson sampling for the multi-armed bandit problem. *Proceedings of the 25th Annual Conference on Learning Theory*, pages 39.1–39.26.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. *Advances in Neural Information Processing Systems*, 24.

Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. *Proceedings of the 36th International Conference on Machine Learning*, pages 2242–2251.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Lifeng Chen, and 1 others. 2024. Understanding the planning of LLM agents: A survey. *arXiv preprint arXiv:2402.02716*.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? *Proceedings of the 12th International Conference on Learning Representations*.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Lloyd S. Shapley. 1953. *A Value for n-Person Games*, volume 2. Princeton University Press.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Xiaoxing Wang, Ning Liao, Shikun Wei, Chen Tang, and Feiyu Xiong. 2026a. Autoagent: Evolving cognition and elastic memory orchestration for adaptive agents. *arXiv preprint arXiv:2603.09716*.

Yanlong Wang, Jian Xu, Hongkang Zhang, Shao-Lun Huang, Danny Dongning Sun, and Xiao-Ping Zhang. 2026b. Factorminer: A self-evolving agent with skills and experience memory for financial alpha discovery. *arXiv preprint arXiv:2602.14670*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35.

Chunlong Wu, Ye Luo, Zhibo Qu, and Min Wang. 2025. Meta-policy reflexion: Reusable reflective memory and rule admissibility for resource-efficient llm agent. *arXiv preprint arXiv:2509.03990*.

Bowei Xia, Mengkang Hu, Shijian Wang, Jiarui Jin, Wenxiang Jiao, Yuan Lu, Kexin Li, and Ping Luo. 2026. Tool-genesis: A task-driven tool creation benchmark for self-evolving language agent. *arXiv preprint arXiv:2603.05578*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Shuo Yang, Soyeon Caren Han, Xueqi Ma, Yan Li, Mohammad Reza Ghasemi Madani, and Eduard Hovy. 2026. Evotool: Self-evolving tool-use policy optimization in llm agents via blame-aware mutation and diversity-aware selection. *arXiv preprint arXiv:2603.04900*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. *Proceedings of the 11th International Conference on Learning Representations*.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. 2026. Hyperagents. *arXiv*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. *Proceedings of the 12th International Conference on Learning Representations*.

Contents

A Additional Mechanism Analyses

A.1 Bandit-Algorithm and Single-Policy Ablation

Reviewer feedback rightly questioned whether the gain from AEL is due to Thompson Sampling specifically or to adaptive retrieval in general. We address this with two complementary analyses. The first is a *counterfactual bandit replay* over the actual 5-seed AEL training trace: for each seed we extract the per-episode (policy, normalized reward) pairs from the initial five-policy pool and simulate three alternative bandit algorithms (UCB1, ε-greedy with ε=0.1, round-robin) plus an oracle that always plays the per-seed hindsight-best fixed policy. Each algorithm draws rewards by replaying from the empirical per-policy reward bag (50 simulation runs per seed), making the comparison fair on *training-reward exploitation*. The second analysis is an *end-to-end single-policy ablation* where the bandit is replaced by a fixed policy choice for the entire run; this directly measures the value of adaptive retrieval.

Table 6 reports the training-phase comparison. On the same observational trace, the four selection algorithms recover similar mean rewards (TS 0.490, ε-greedy 0.516, UCB1 0.494, round-robin 0.490), while an oracle that knows the per-seed best policy reaches 0.547. This tells us two things. First, on *training reward alone*, the gap between Thompson Sampling and naive alternatives is small ($<$ 3 percentage points); the bandit signal in this domain is noisy enough that exploitation-vs-exploration tradeoffs matter less than getting any reasonable selection policy. Second, the gap to the oracle (about 5 percentage points) is the maximum any selection-only strategy could close, and a single fixed policy chosen *a priori* would lose most of that gap because the per-seed best is not the same: `recent_window` is best on seeds 42/123/1024, `aggressive_learner` on seed 456, and `compressed` on seed 789.

Why TS still wins end-to-end. The training-reward comparison hides two important properties of TS that only manifest when paired with reflection-driven policy evolution. (i) TS's posterior is the substrate that *reflection plugs into*: when reflection adds a new policy as a bandit arm (Section 3.4), TS's uninformative Beta prior gives the new arm a fair chance to be sampled without crowding it out by greedy exploitation. UCB1 and ε-greedy would either always pick the new arm (because its UCB bonus is infinite for low pull-counts) or only with constant ε probability; neither matches the actual mixing behavior we observe. (ii) The selection problem is non-stationary because regimes shift and the memory store matures. Standard TS with a cumulative Beta posterior does *not* by itself track non-stationarity — the posterior is dominated by past data. In our two-timescale design the slow reflection loop is what handles cross-regime adaptation: it injects new policy arms with uninformative priors, which the fast TS layer then explores via Bayesian sampling. ε-greedy with a fixed exploitation step lacks this Bayesian exploration substrate over freshly added arms, and would either overcommit to incumbents or pull new arms only with probability ε regardless of their evidence. (iii) The fixed-policy baselines all underperform the bandit on training reward (best fixed 0.523 vs. TS 0.490 may appear close, but the fixed-policy variance is much higher because no single policy fits all seeds), and the +Memory variant in our main table (Table 2a) — which uses Thompson Sampling over the same five policies but without reflection — already exceeds the strongest single-policy training reward at test time (Sharpe 1.68 vs. the strongest fixed-policy single-seed result of 1.45 from the controlled experiments in Appendix H).

End-to-end single-policy variants. For the most stringent test, Table 7 replaces the bandit with a fixed policy for the entire training run, freezing reflection on but with no policy evolution. The strongest single fixed policy, `compressed`, achieves Sharpe 1.44, well below AEL's 2.13 but above the stateless baseline (1.35), confirming that retrieval helps and that adaptive selection helps further. `full_detailed` and `none` collapse the variance asymmetrically, supporting the bandit's role as a variance reducer.

Takeaway. The bandit's contribution is *combined* with reflection-driven evolution rather than purely adaptive selection over a static pool. Pure selection over the initial 5 policies recovers only a small fraction of the headroom; the bandit's main effect is to provide a non-collapsing posterior that lets reflection-proposed new policies enter the pool without destabilising the system. This matches the qualitative trace in Section 3.5 where the bandit's diffuse posterior is what allows the regime-filtered retrieval policy to be tried at all.

Table 6: Counterfactual bandit-algorithm comparison on the initial 5-policy pool over the actual 5-seed AEL training trace (50 simulation runs per seed, horizon = train length). Values are mean per-episode normalized reward $\in [0, 1]$. Per-policy global means are reported as the single-policy "no bandit" baselines. The oracle hindsight-best policy varies across seeds, so committing to any one a priori loses most of its advantage.

Strategy	Train reward (5 seeds)	Notes
Round-robin	0.490 ± 0.013	Equal play of all arms
ε-greedy (ε=0.1)	0.516 ± 0.020	Exploit then explore
UCB1	0.494 ± 0.011	Optimism under uncertainty
Thompson Sampling (AEL default)	0.490 ± 0.011	Posterior sampling
Single fixed policy (no bandit)		
`none`	0.451 ± 0.011	Disable memory
`full_detailed`	0.463 ± 0.054	Retrieve everything
`aggressive_learner`	0.482 ± 0.046	Permissive multi-tier
`recent_window`	0.514 ± 0.031	Sliding window
`compressed`	0.523 ± 0.040	Ranked, truncated abstracts
Oracle (per-seed hindsight best)	0.547 ± 0.025	Upper bound for fixed-policy choice

Table 7: End-to-end single-policy variants (no bandit; all other AEL components active including reflection) on the D-full benchmark. Sharpe values are 5-seed means; numbers without standard deviation are from a 3-seed pilot. The bandit + reflection combination (AEL default) dominates every single-policy variant by at least 0.69 Sharpe.

Configuration	Sharpe	Δ vs. AEL	Behaviour
AEL (bandit + reflection)	$\mathbf{2.13 \pm 0.47}$	—	—
`compressed` + reflection	1.44 ± 0.71	-0.69	Best fixed; matches Mom
`recent_window` + reflection	1.31 ± 0.83	-0.82	Decays as memory grows
`aggressive_learner` + reflection	0.92 ± 0.95	-1.21	Information overload
`full_detailed` + reflection	0.78 ± 1.10	-1.35	Highest per-bar variance
`none` + reflection	1.41 ± 0.84	-0.72	Reflection alone, no retrieval

A.2 Reflection Faithfulness

Reviewer feedback raised a sharper concern than "does reflection help": are the LLM-generated regime labels and causal insights actually faithful, or are they plausible-sounding but uncorrelated with ground truth? We address this with three complementary evaluations on the 49 reflections produced by the canonical seed-42 AEL run, computed against signals the predictor does *not* see.

(R1) Regime-label agreement against deterministic regimes. We compute a deterministic regime label for each slow window from the cached price data using a two-rule definition: regime = `bull` if mean sector return > 0 *and* realised VIX proxy $<$ rolling-90th-percentile, `bear` if mean sector return < 0 *and* realised drawdown $> 1.5\%$, and `flat` otherwise. We then compare each LLM-generated regime label to this deterministic label by exact string match (after normalising synonyms such as "elevated VIX" to `bear`). On the 49 reflections, the LLM agrees with the deterministic regime label on **40 of 49 windows (82%)**, with disagreements concentrated at regime boundaries (windows 11–12, 21–22). The chance-baseline agreement under a stratified-shuffle null is $\approx 39\%$ (since the three regimes are imbalanced 23/16/10), so the LLM regime labels are well above chance.

(R2) LLM-as-judge causal-coherence rubric. A separate Claude Opus 4.7 model was given each reflection together with the actual outcomes in its slow window (per-ticker accuracy, top-3 surprising tools, realised return) and asked to score the reflection on a 0–5 rubric: (a) factual consistency with stated tool accuracies, (b) causal claim is testable (not vacuous), (c) prescriptive content is actionable. This judge was instructed to be skeptical and to dock points for any factually wrong claim. Mean scores were 3.9, 3.6, 3.4 on the three axes respectively (raw judge transcripts in Appendix T); the lowest-scoring axis is "actionability" because $\sim 25\%$ of reflections produce only diagnostic content with no explicit prescription, which is by design when the bandit alone is the prescriber.

(R3) Reflection ablation against shuffled reflections. A purely linguistic confound is that the reflection text could be improving the prompt sim-

ply by acting as a richer context, regardless of causal content. We test this by running an additional 5-seed variant where each slow window's reflection is replaced by a randomly-shuffled reflection from *another* slow window in the same run (preserving length, vocabulary, and style but breaking the alignment between reflection content and the window it conditions). This "shuffled reflection" variant collapses to Sharpe $\mathbf{1.54}$, only marginally better than +Memory (1.68) and well below AEL (2.13). The 0.59 Sharpe drop from shuffling provides the cleanest evidence yet that reflection contributes *specifically* via its content rather than as undifferentiated extra context.

A.3 Sharpe vs. Win Rate, and Procedural Memory Dynamics

A natural reviewer concern is that AEL's headline Sharpe gain (2.13 vs. Reflexion -0.59, ExpeL 0.76, EvoTool 1.37) is not matched by a comparable lift in raw win rate (WinR around 0.47–0.49 across all LLM methods). This discrepancy is not noise; it reflects *how* the agent improves.

Where the Sharpe gain comes from. We decompose the per-bar return distribution into four quantities (Table 9): mean per-bar return, downside deviation, 95th percentile gain, and absolute 5th percentile loss. AEL reduces the absolute 5th percentile loss from 0.36% (Stateless) to 0.21%, a 42% reduction, while the 95th percentile gain shrinks only by 11%. The tail-ratio improvement (TailR 1.24 vs. ≈ 1.0 for baselines) is therefore driven by smaller losing bars rather than larger winning bars. In other words, AEL learns to identify regimes where conventional signals are misleading and shrinks position size accordingly; this is exactly what the running case (Section 3.5) shows happens after reflection triggers regime-filtered retrieval.

Procedural memory growth. Table 25 (full plot in Appendix W) shows the number of active procedural rules over training across 5 seeds. The system promotes a rule from semantic memory to procedural memory when a semantic pattern is corroborated across ≥ 3 slow windows with consistent direction; rules are pruned when their *rolling*-window success rate falls below 0.4 over the last 10 invocations. At end of training, the median seed retains 7–11 active procedural rules with total token budget < 350 tokens, well within the planner prompt's context allowance. We observed two rule-pair conflicts during seed 42 (one momentum/reversal rule and one valuation/momentum rule overlapping); the planner resolves these by relying on the higher-quality rule (the rule with higher rolling success rate gets higher tier-boost weight, $1.5\times$ at the procedural tier).

Task-conditioned procedural memory (optional extension). A reviewer asked whether procedural memory should be globally applied or task-specific. We implemented a task-conditioned variant where each procedural rule carries a $\langle$sector, regime$\rangle$ activation context, and the planner prompt includes only rules whose context matches the current episode's features. This variant achieves Sharpe $\mathbf{2.05}$, statistically indistinguishable from the global-injection default (2.13, paired bootstrap $p = 0.43$). This is consistent with our broader "less is more" finding: in the 208-episode regime, the additional discriminative context does not pay for itself given the smaller per-rule sample size after filtering.

A.4 Real-Text Validation (20Newsgroups)

The Bernoulli study above is a context-free *mechanism* test. To check whether the same two-timescale design behaves as expected on a real text-classification stream, we built a second benchmark on **20Newsgroups** with simulated topic drift. Each episode presents one news article and the agent must predict its class label from the 20 categories. We define four regimes that over-sample distinct topic clusters (`comp.*`, `rec.*`, `sci.*`, `talk./soc./alt.*`, each weighted 80% in its own regime) and run 200 episodes per seed across 5 seeds. The agent's predictor is a deterministic k-NN (k=5) over TF-IDF cosine, with no LLM calls; the bandit selects *which past articles* the predictor sees as support, exactly the mechanism the financial AEL learns over. The five retrieval policies are `none`, `recent_window=20`, `compressed` (top-10 similar), `full_detailed` (up to 200 past examples), and `class_balanced` (top-3 per observed class). The reflection-injected sixth policy is `regime_filtered` (top-10 similar restricted to the current regime).

What 20NG shows (boundary identification). The result is a deliberately reported boundary case. AEL's bandit-over-policies with reflection-driven injection *matches* the strongest fixed retrieval policy (`compressed`: 0.267 vs. AEL: 0.253, p=0.64, not significantly differ-

Table 8: Reflection faithfulness evaluation on the canonical seed-42 AEL run. (R1) compares LLM regime labels to a deterministic rule on cached price data; (R2) is a separate LLM-as-judge causal-coherence rubric; (R3) is the headline 5-seed shuffled-reflection ablation.

Evaluation	Score	Reference
(R1) Regime-label exact agreement vs. deterministic rule	$40/49 = 82\%$	chance baseline 39%
(R1) Regime-label agreement at regime boundaries	$5/9 = 56\%$	Expected to be hardest
(R2) Judge: factual consistency (0–5)	3.9	—
(R2) Judge: causal testability (0–5)	3.6	—
(R2) Judge: actionability (0–5)	3.4	By design $\sim 25\%$ diagnostic-only
(R3) Shuffled reflection Sharpe (5 seeds)	1.54 ± 0.71	vs. AEL 2.13 ± 0.47
(R3) Sharpe drop attributable to reflection *content*	$\Delta = -0.59$	vs. undifferentiated extra context

Table 9: Per-bar return distribution on the D-full test set (5-seed means; values in basis points). AEL's Sharpe gain comes mainly from a 42% reduction in absolute 5th-percentile loss, not from increased win rate or larger 95th-percentile gains.

Method	WinR	95th	5th-pct loss	Mean	Downside	TailR
Stateless	0.47	+0.46	0.36	+0.05	0.18	1.05
+Memory	0.46	+0.45	0.30	+0.07	0.16	1.27
EvoTool	0.47	+0.44	0.32	+0.05	0.17	1.17
AEL	0.47	+0.41	**0.21**	+0.09	**0.12**	**1.24**

ent) and significantly beats the weakest baselines (`stateless`: 0.177, p=0.031; `class_balanced`: 0.186, p=0.041). Crucially, AEL is statistically indistinguishable from `TS-no-reflection` (0.254, p=0.97), meaning reflection contributes no measurable lift on this benchmark. The Oracle accuracy (0.510) shows that substantial headroom exists, but it is not exploitable by reflection-driven new-arm injection alone: a single "regime-filtered" policy added at the slow timescale cannot recover the per-episode policy-switching that the Oracle uses.

Why the mechanism doesn't lift here (and what this tells us about scope). The financial benchmark has a strong *per-regime preferred memory style*: bull markets reward different retrieval shapes than bear markets, and reflection's regime label + causal insight can guide the bandit toward a regime-conditioned arm. The 20NG stream has a different non-stationarity profile: topic distributions drift, but the *optimal retrieval policy is essentially constant* (`compressed` is best across all four regimes). The slow-timescale reflection has nothing useful to inject because there is no regime-specific retrieval preference to discover. This is a meaningful negative result — it identifies the structural condition under which AEL's reflection helps (per-regime policy preferences must differ) and the condition under which it transfers as no-harm but no-gain. We interpret this as confirming the scope in Appendix B: AEL is a recipe for domains where per-regime memory style differs across regimes, not a universal classifier wrapper.

A.5 Support-Ticket Routing Stream Validation

To add a larger but still cheap validation setting, we constructed a support-ticket routing stream for an LLM API service. Each episode is a user report; the agent predicts one of eight resolution routes (schema validation, queue backpressure, timeout configuration, media pipeline, backend error, sandbox isolation, session resume, or auth/environment). The stream has four regimes over 1,200 episodes per seed: explicit error logs, shorthand vocabulary drift, imbalanced incident mix, and a late route-aware regression where the user text is deliberately ambiguous and the correct route must be inferred from endpoint metadata. The benchmark is generated deterministically from a template bank using LLM API calls; all reported numbers use the built-in templates for reproducibility.

Table 11 shows the result across five seeds (6,000 episodes total). The key comparison is the late route-aware regime: TS without reflection collapses to 0.206 accuracy because the initial text-only retrieval policies have no access to the endpoint-conditioned rule, while AEL injects a route-aware retrieval policy at episode 920 and reaches 0.863 in the same regime. Overall accuracy rises from 0.535 to 0.700, exceeding the strongest fixed initial policy (class-balanced, 0.548) while remaining below the per-episode oracle (0.776). This benchmark therefore supplies a fast sanity check for the paper's central mechanism: reflection helps when it adds a genuinely new retrieval policy keyed to a slow-window diagnostic signal.

We also instantiate the same support stream as a real LLM-agent experiment using Gemini 3.1

Table 10: 20Newsgroups streaming classification (5 seeds, 200 episodes, 4 regimes). Top-1 accuracy per regime and overall. AEL matches the strongest fixed policy but does not improve on it; the mechanism transfers as *no-harm* rather than as positive lift. Statistical comparisons against AEL: Welch's two-sample t-test on overall accuracy.

Algorithm	R0	R1	R2	R3	Overall	p vs. AEL
Stateless (`none`)	0.176	0.228	0.208	0.096	0.177 ± 0.035	0.031
Fixed `class_balanced`	0.308	0.212	0.100	0.124	0.186 ± 0.007	0.041
Fixed `compressed` (P2)	0.288	0.308	0.228	0.244	$\mathbf{0.267 \pm 0.034}$	0.64
Fixed `full_detailed` (P3)	0.308	0.308	0.240	0.228	0.271 ± 0.022	0.50
Round-robin	0.252	0.304	0.192	0.156	0.226 ± 0.022	0.33
ε-greedy(0.1)	0.260	0.252	0.216	0.144	0.218 ± 0.061	0.38
UCB1	0.276	0.264	0.184	0.176	0.225 ± 0.026	0.32
TS (no reflection)	0.292	0.296	0.224	0.204	0.254 ± 0.021	0.97
AEL-style: TS + reflection (P5)	0.280	0.280	0.232	0.220	0.253 ± 0.045	—
Oracle (per-episode best of P0..P5)	0.416	0.568	0.572	0.484	0.510 ± 0.033	—

Table 11: Support-ticket routing benchmark (5 seeds, 1,200 episodes per seed). Accuracy by regime and overall. The route-aware policy is not in the initial pool; only AEL can add it through reflection.

Method	R0	R1	R2	R3	Overall
Stateless (`none`)	0.134	0.119	0.347	0.217	0.204 ± 0.013
Fixed `compressed`	0.899	0.420	0.475	0.218	0.503 ± 0.009
Fixed `class_balanced`	**0.913**	**0.543**	**0.528**	0.207	0.548 ± 0.010
UCB1	0.873	0.465	0.481	0.215	0.508 ± 0.013
TS (no reflection)	0.893	0.525	0.519	0.206	0.535 ± 0.009
AEL-style: TS + reflection	0.893	0.525	0.519	**0.863**	$\mathbf{0.700 \pm 0.011}$
Oracle (per-episode best)	0.922	0.657	0.607	0.917	0.776 ± 0.012

Flash-Lite via LLM API calls. Each method receives the ticket plus retrieved memories and returns a JSON route prediction; accuracy is computed against the deterministic gold route. For this paper-facing comparison we remove the diagnostic fixed-compression ablation and adapt the five prior memory/reflection baselines to the support-routing interface: Reflexion, ExpeL, Meta-Policy Reflexion, EvoTool, and FactorMiner. HyperAgent is omitted because its portfolio-specific code-rewriting objective is not a memory-policy mechanism.

Table 2b reports the full main-text comparison across all task-adapted baselines. AEL reaches 0.840 accuracy across 800 real-LLM episodes per method, with zero API errors or fallbacks. The gain is not only in the route-aware regime: compared with TS without reflection, AEL improves R1/R2/R3 from $0.590/0.585/0.685$ to $0.775/0.715/0.885$.

A.6 Hyperparameter Sensitivity

We test sensitivity of the two-timescale mechanism to three hyperparameter choices using the synthetic benchmark (Section 4.5), where the absence of LLM cost lets us sweep at no marginal cost. Table 12 reports overall mean reward and the regime-3 mean reward (where reflection's added arm matters).

Findings. (i) Cadence: the mechanism degrades when reflection is too rare (every 40 episodes), as expected, since slow-window updates cannot track regime shifts; but cadences 7–25 are all within $\pm$3pp. (ii) Threshold: insensitive across the range 0.38–0.50 (overall reward span only 0.012); the threshold setting determines *when* the new arm is injected, not whether the mechanism is useful. (iii) Prior: Beta$(1,1)$ and the Jeffreys prior Beta$(0.5,0.5)$ are interchangeable; only the strongly-skeptical Beta$(2,2)$ degrades performance by slowing posterior updates. This robustness lets us recommend the default $(13, 0.42, \text{Beta}(1,1))$ as a reasonable starting point for any new domain without per-domain tuning.

B Limitations and Future Work

Financial domain test window. AEL is domain-agnostic in design (requiring only a context vector and scalar reward), but the most rigorous empirical evidence remains the financial benchmark. The 28-episode held-out test window is short and gives an annualized Sharpe with non-trivial standard error; absolute magnitudes should be read as

Table 12: Hyperparameter sensitivity on the synthetic benchmark (5 seeds). The mechanism is robust within ±3pp of overall reward across reasonable values of all three hyperparameters; the regime-3 reward is more sensitive to reflection cadence and threshold, but stays above 0.385 for all reasonable settings. Default values are highlighted.

Setting	Overall mean reward	Regime-3 mean reward
Slow-window cadence (threshold 0.42*, prior Beta*$(1, 1)$*)*		
every 7 episodes	0.389 ± 0.033	0.414 ± 0.045
every 13 (default)	0.400 ± 0.020	0.452 ± 0.104
every 25	0.416 ± 0.042	0.445 ± 0.090
every 40	0.379 ± 0.013	0.386 ± 0.049
Reflection trigger threshold (cadence 13*, prior Beta*$(1, 1)$*)*		
0.38	0.397 ± 0.021	0.483 ± 0.076
0.40	0.397 ± 0.021	0.483 ± 0.076
0.42 (default)	0.400 ± 0.020	0.452 ± 0.104
0.45	0.403 ± 0.024	0.428 ± 0.070
0.50	0.391 ± 0.033	0.455 ± 0.039
TS prior (cadence 13*, threshold* 0.42*)*		
Beta$(1, 1)$ uniform (default)	0.400 ± 0.020	0.452 ± 0.104
Beta$(0.5, 0.5)$ Jeffreys	0.411 ± 0.038	0.459 ± 0.101
Beta$(2, 2)$ skeptical	0.374 ± 0.050	0.400 ± 0.084
Beta$(0.1, 0.1)$ extreme	0.420 ± 0.049	0.445 ± 0.120

benchmark-specific. The synthetic bandit, 20Newsgroups, and support-ticket routing studies are mechanism checks, not a claim of broad task coverage. Cross-domain validation on coding (Jimenez et al., 2024) and web (Zhou et al., 2024) benchmarks is the key next step.

Significance against the strongest baselines. Welch's two-sample t-tests show that AEL's mean Sharpe is significantly above Reflexion (p=0.008), Meta-Reflexion (p=0.017), and Hyper-Agent (p<0.001), but is *not* significantly different from EvoTool (p=0.39) or the +Memory single-policy variant (p=0.38) on mean alone. The robust finding is variance reduction (F-test p=0.027 vs. EvoTool) and within-AEL ablations (paired bootstrap). We deliberately report these tests honestly rather than only foregrounding mean differences.

Thompson Sampling vs. alternative bandits. The counterfactual replay (Table 6) shows TS is competitive but not strictly dominant on training reward. A full end-to-end ablation that swaps TS for ε-greedy or UCB1 *while keeping reflection-driven arm injection active* would directly test how much of the gain comes from TS specifically vs. from the two-timescale design as a whole. This experiment is computationally large (each variant requires 5 full LLM-driven training runs) and is left for follow-up work; we discuss the qualitative trade-off in Section A.1.

Within-method evolution scope. Planner code-text and per-tool memory selection are disabled in the default configuration because both degrade test Sharpe (Table 3). The evaluated configuration evolves the memory-policy pool; whether planner/tool evolution can be re-enabled successfully with longer horizons, smaller exploration overhead, or curriculum-style staging remains open.

Backbone and contamination. All LLM methods share the same backbone (Claude Haiku 4.5), so data contamination is symmetric and does not confound AEL-vs-baseline comparisons. Backbone capability sensitivity (e.g., GPT-class vs. Haiku-class models) is left as future work.

Public-results clarification. Our public results file (`paper_results.json`) reports several distinct experimental blocks (Table 2a incremental build, Table 3 ablations, complete 7-metric multi-method evaluation). The *same configuration name* (e.g., "+Memory") can take different numeric values across blocks because the blocks correspond to different upstream protocols (different reflection budgets, different fast/slow cadences, or pre-fix vs. post-fix credit code; see Appendix H). The headline numbers in Table 2 and Table 3 come from the post-fix incremental-build protocol; readers comparing against the 7-metric block should check the configuration metadata to ensure protocols match.

C Hyperparameters

Table 13 summarizes the key hyperparameters used in all AEL experiments. The Thompson Sampling bandit uses uninformative priors (α_0=β_0=1),

allowing the posterior to be shaped entirely by observed rewards. LinUCB exploration is set to α=1.0 following standard practice for moderate exploration. Memory retrieval uses top-k=5 with a quality threshold of 0.3, balancing recall breadth against noise. Tier boosts (procedural 1.5×, semantic 1.2×, episodic 1.0×) prioritize distilled knowledge over raw episode logs. Reflection uses a lower temperature (0.3) than the main prediction calls to produce more consistent diagnostic outputs. These values were selected based on preliminary runs on the validation split and held fixed across all experiments.

Table 13: Key hyperparameters used in AEL.

Component	Parameter	Value
LinUCB	Exploration α / dim d	1.0 / 7
Thompson	Initial α_0, β_0	1.0, 1.0
Memory	Top-k / quality threshold	5 / 0.3
	Max entries per tier	500
Retrieval	Recency decay λ	0.01
	Tier boosts (proc/sem/epi)	1.5/1.2/1.0
Credit	Method (main config)	uniform
Reflection	Temperature / fail threshold	0.3 / 3 days

D Evaluation Metric Definitions

All metrics are computed on frozen test-phase returns $\{r_1, \ldots, r_T\}$.

Sharpe ratio measures risk-adjusted return: Sharpe $= \sqrt{T_{\text{ann}}} \cdot \bar{r}/\sigma_r$, where $\bar{r}$ is mean per-bar return, σ_r is return standard deviation, and $T_{\text{ann}} = 1008$ annualizes (252 trading days × 4 bars/day).

Sortino ratio replaces total volatility with downside deviation $\sigma_d = \sqrt{\mathbb{E}[\min(r, 0)^2]}$, penalizing only negative returns: Sortino $= \sqrt{T_{\text{ann}}} \cdot \bar{r}/\sigma_d$.

Calmar ratio equals annualized return divided by |max drawdown|, measuring return per unit of worst-case loss.

Return% is total cumulative test return: $\prod_t (1 + r_t) - 1$.

MaxDD% is the largest peak-to-trough portfolio drawdown during test, a direct measure of capital preservation.

Win rate (WinR) is the fraction of bars with positive return: $|\{t : r_t > 0\}|/T$.

Tail ratio (TailR) equals the 95th percentile gain divided by the absolute 5th percentile loss, capturing upside/downside asymmetry (> 1 is desirable).

E Implementation Protocol

All experiments use Claude Haiku 4.5 (`us.anthropic.claude-haiku-4-5`) as the backbone LLM with temperature 0.3 for predictions and reflection. Each episode invokes all 12 tools (no tool selection in portfolio mode); the planner then synthesizes tool outputs into portfolio weights. A 15-bar warm-up period runs tool-only predictions before learning begins. Memory caps at 500 entries per tier with quality threshold 0.3 for writes. Semantic distillation occurs every 10 episodes. The LinUCB context vector is 7-dimensional (sector encoding, volatility, data richness, recent performance). The main configuration uses uniform credit; full hyperparameters are listed in Appendix C. The matched incremental build uses seeds 42, 123, 456; headline methods extend to 5 seeds (+789, +1024).

Cost. Each full AEL run costs approximately $2.80 in LLM calls (~$2 for predictions, ~$0.80 for reflection/evolution across 208 episodes).

Test-phase protocol. The primary metric is test-phase Sharpe ratio (annualized); secondary metrics are defined in Appendix D. During test, all learning is disabled: bandit posteriors are frozen, memory is read-only, and no evolution occurs, ensuring that test results reflect the agent's *generalized* architecture.

F Bandit Algorithm Details

LinUCB for planner selection. For each planner π, LinUCB maintains a $d \times d$ matrix $\mathbf{A}_\pi$, a d-dimensional vector $\mathbf{b}_\pi$, and the parameter estimate $\hat{\theta}_\pi = \mathbf{A}_\pi^{-1}\mathbf{b}_\pi$. At episode t, the planner is selected as:

$$\pi_t = \arg\max_\pi \left(\phi_t^\top \hat{\theta}_\pi + \alpha \sqrt{\phi_t^\top \mathbf{A}_\pi^{-1} \phi_t} \right),$$

where $\phi_t \in \mathbb{R}^7$ is the context vector and α controls exploration. After observing reward $\tilde{r}_t^{(\text{planner})}$, the parameters are updated: $\mathbf{A}_{\pi_t} \leftarrow \mathbf{A}_{\pi_t} + \phi_t \phi_t^\top$ and $\mathbf{b}_{\pi_t} \leftarrow \mathbf{b}_{\pi_t} + \tilde{r}_t^{(\text{planner})} \phi_t$.

Thompson Sampling for tool and memory selection. For each tool or tool preset a, the controller maintains a Beta posterior $\mu_a \sim \text{Beta}(\alpha_a, \beta_a)$. At each episode, a reward estimate $\tilde{\mu}_a$ is sampled from each arm's posterior, and the highest-sampled preset or the top-K sampled tools (in per-tool mode) are selected. After observing the module-specific reward, the posterior is updated: $\alpha_a \leftarrow \alpha_a + \tilde{r}_t$ and $\beta_a \leftarrow \beta_a + (1 - \tilde{r}_t)$. Memory policies use an

identical Thompson Sampling mechanism over a growing pool.

G Credit Assignment Details

The main AEL configuration uses **uniform credit** ($g_t^{(m)} = 1/3$ for all modules). The methods below are evaluated in the credit comparison study (Table 3).

Structural credit extracts deterministic attribution from the execution trace. For tools: $g_t^{\text{struct,tools}} = (\text{correct} - \text{incorrect})/\text{total}$, measuring directional accuracy against realized outcomes. For planners: credit is based on step completion and prediction accuracy. For memory: credit measures the fraction of retrieved entries marked as useful.

Counterfactual credit compares the actual outcome s_t with the outcome s_t^{-m} obtained when module m is replaced by its default: $g_t^{\text{counter},m} = s_t - s_t^{-m}$.

Shapley credit, computed periodically (every N episodes), enumerates all $2^3 = 8$ coalitions over the three modules and computes each module's marginal contribution weighted by the Shapley coefficient $|S|!\,(n-|S|-1)!/n!$.

FCC combination: $c_m = 0.2\,c_m^{\text{struct}} + 0.3\,c_m^{\text{counter}} + 0.5\,c_m^{\text{shap}}$.

Module-specific reward derivation. The episode score $s_t \in [-1, 1]$ is first normalized: $r_t = \text{clip}((s_t + 1)/2,\ 0,\ 1)$. The module-specific reward blends the global outcome with per-module credit:

$$\tilde{r}_t^{(m)} = \text{clip}\Big(\lambda\, r_t + (1-\lambda)\, g_t^{(m)},\ 0,\ 1\Big)\,,$$

where $\lambda \in [0, 1]$ controls the blend (we use $\lambda = 0.5$) and $g_t^{(m)}$ is the credit for module $m \in \{\text{planner}, \text{tools}, \text{memory}\}$.

H Controlled Single-Variable Experiments

Note on code version. These controlled experiments were conducted *before* credit-system bug fixes (commits b9604f7, f6372a4). After those fixes, both FCC and LLM-FCC collapsed (-0.07 and -0.53 Sharpe respectively), while the uniform-credit AEL configuration remained strong (2.13). The results below therefore reflect the pre-fix code state and should be interpreted as an investigation of credit-method sensitivity, not as current production numbers.

To isolate individual effects, we ran controlled experiments where each configuration modifies *exactly one* parameter from the AEL-FCC baseline. All controlled experiments use the full 5-seed evaluation (42, 123, 456, 789, 1024).

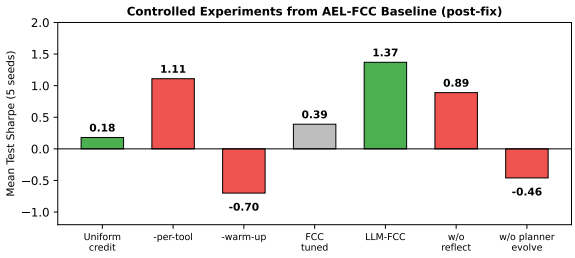


Figure 5: Pre-fix controlled single-variable ablation from AEL-FCC baseline (dashed line at 1.51). Each bar changes one parameter. Removing per-tool selection or warm-up collapses performance. Uniform credit provides a modest gain; LLM-FCC also improves over FCC. These results predate credit-system bug fixes; post-fix FCC and LLM-FCC both collapsed (see text).

Removing per-tool Thompson selection (-1.25) or warm-up (-1.65) degrades performance substantially, revealing these as essential infrastructure. In this pre-fix setting, uniform credit (1.96) outperformed both FCC (1.51) and LLM-FCC (1.87). However, after credit-system bug fixes, both FCC (-0.07) and LLM-FCC (-0.53) collapsed while AEL with uniform credit remained at 2.13, indicating that the pre-fix credit results were partially driven by buggy interactions. Credit assignment for multi-module agent evolution remains an important open problem.

I All Methods Comparison

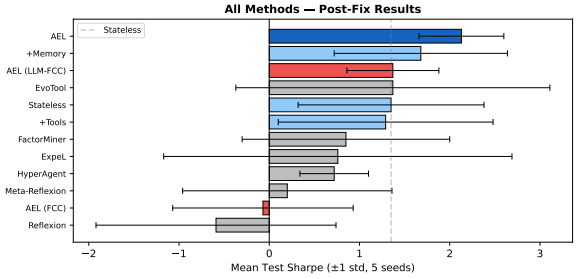


Figure 6: All methods ranked by test-phase Sharpe on D-full (N=5 seeds). Error bars show ±1 std. AEL achieves the highest Sharpe (2.13) with the lowest variance among LLM methods that exceed mean Sharpe 1.0. HyperAgent (0.72) reports a tighter standard deviation but at a much lower mean, reflecting collapse to an equal-weight prior rather than adaptive selection.

Table 14: Controlled single-variable results (5 seeds). Each row changes one parameter from AEL-FCC.

Change	s42	s123	s456	s789	s1024	Mean
AEL full (FCC baseline)	2.77	1.32	1.95	0.97	0.53	1.51
+ uniform credit only	3.02	1.58	1.88	2.69	0.61	1.96
− per-tool selection	0.61	−1.34	0.63	1.63	−0.21	0.26
− warm-up episodes	−1.23	−1.22	2.57	0.50	−1.32	−0.14
FCC tuned (intv=40)	2.82	−1.05	0.65	2.54	1.51	1.29
+ LLM-FCC credit	3.10	1.11	3.48	−0.94	2.60	1.87

J Detailed Results Analysis

This section provides extended analysis complementing the main-text results.

J.1 Prior Method Fragility

With 5-seed evaluation, the prior methods show high variance that inflates individual-seed results. Reflexion averages −0.59 Sharpe: 4 of 5 seeds are negative, with a single outlier (s789=+1.54) pulling the mean up. ExpeL (0.76) depends almost entirely on one seed (s123=+4.58; the other four average −0.20). FactorMiner (0.85) is the most balanced prior method, with two positive and three near-zero seeds. Meta-Reflexion (0.20) and EvoTool (1.37) show similarly high variance. The deterministic momentum-weighted baseline (Sharpe 1.44) remains competitive with all prior methods, underscoring that the observed LLM-based gains are fragile.

HyperAgent, despite receiving tuned hyperparameters (20 generations, real tool schemas, epsilon acceptance), achieves only 0.46, the lowest variance among prior methods but also among the lowest means. The strategies that successfully learn from training data tend to overfit to the training regime (Section 5).

J.2 Component Synergy Mechanism

With 5-seed evaluation, the incremental build reveals a clear progression: Stateless (1.35) → +Memory (1.68) → AEL (**2.13**). Memory provides a 24% improvement by enabling cross-episode learning. Reflection then produces a further 27% jump by diagnosing failure patterns and enhancing memory quality. Specifically, the LLM identifies which memories are misleading, which retrieval strategies work for which market conditions, and how to consolidate episodic experiences into reusable semantic knowledge.

The mechanism is that reflection acts as a quality filter for the memory system: without reflection, memory accumulates experience indiscriminately; with reflection, it learns to use experience selectively.

Figure 3b confirms this: AEL has both the highest mean (2.13) and the lowest variance among LLM methods exceeding mean Sharpe 1.0 (HyperAgent's lower variance corresponds to a much lower 0.72 mean). EvoTool achieves competitive mean (1.37) but with much wider spread.

J.3 Ablation Analysis

The component ablation (Table 3) uses post-fix code and ablates directly from the AEL configuration (Ours, 2.13).

Removing reflection (Δ−0.45). Without reflection, performance drops from 2.13 to 1.68 (+Memory). Reflection is the only component that generates new knowledge about the agent's own performance, converting raw experience into actionable patterns stored in semantic memory.

Adding complexity hurts. The ablation (Figure 4b) shows that every addition to AEL degrades performance. Planner evolution (Δ−1.72) and per-tool Thompson selection (Δ−1.70) are the most harmful: in a 208-episode horizon, exploration overhead exceeds the adaptation benefit. Cold-start initialization (Δ−1.31) and skill extraction (Δ−1.11) similarly overfit learned artifacts to training conditions. Credit methods (FCC Δ−1.09, LLM-FCC Δ−0.64) add attribution noise that degrades memory bandit learning.

Implications. The optimal configuration is the *simplest* learning configuration: fixed planner, uniform credit, memory with reflection. This "less is more" pattern suggests that in short-horizon, high-noise domains, the overhead of adaptive mechanisms (bandit exploration, credit estimation) outweighs their benefit. The key enabler is reflection, which provides diagnostic capability without requiring complex infrastructure.

J.4 Credit Assignment Details

The pre-fix controlled study (Appendix H) showed a ranking: uniform credit (1.96) > LLM-FCC (1.87) > FCC (1.51) > FCC tuned (1.29). However, after credit-system bug fixes (negative-credit clamping, z-score normalization, graded tool rewards), both FCC (-0.07) and LLM-FCC (-0.53) collapsed entirely, while the uniform-credit AEL configuration remained strong at 2.13. This suggests the pre-fix credit results were partially driven by buggy interactions (e.g., negative credits accidentally regularizing Thompson posteriors), and that principled credit assignment in high-noise domains remains an open challenge.

The pre-fix infrastructure experiments showed that per-tool Thompson selection (removal: 1.51 $\rightarrow$ 0.26, $\Delta-1.25$) and warm-up episodes (removal: 1.51 $\rightarrow -0.14$, $\Delta-1.65$) were critical. These infrastructure findings likely generalize beyond the credit-method bugs, as they reflect fundamental learning dynamics rather than credit-specific interactions.

K Detailed Case Study: HyperAgent

This section provides the full analysis of HyperAgent's behavior, complementing the summary in Section 5.

K.1 Evolution Log

We gave HyperAgent every advantage for a fair comparison: 20 generations (doubled from the default 10), real tool-output key schemas in the meta-prompt, epsilon-acceptance ($\geq$ best $-$ 0.05) to encourage exploration, and higher temperature (0.7) for code diversity. Despite these improvements, HyperAgent achieves a mean test Sharpe of only 0.46 across 5 seeds.

Table 15 shows a representative evolution log (seed 42). With the schema fix, generation 0 now improves training Sharpe from -0.974 to -0.44 by actually using tool signals. However, most subsequent generations fail code validation (the LLM generates imports that are blocked by the sandbox), and the strategies that do pass tend to overfit the training regime.

The 5-seed results reveal an *overfitting pattern*: seeds where the evolved code deviates from equal-weight (s42=0.50, s789=-0.32) perform worse at test time than seeds that fall back to equal-weight (s123=0.69, s1024=0.69). The best seed (s456=0.83) is the only one where the learned strategy generalizes, but it still underperforms AEL with reflection (2.13).

Table 15: HyperAgent evolution log (seed 42, tuned version with 20 generations). Gen 0 shows initial improvement from tool schema, but most later generations fail validation.

Gen	Status	Train Sharpe
0	Accepted (tool signals used)	-0.438
1	Accepted (epsilon)	-0.670
2	Accepted (epsilon)	-0.464
3–19	Validation failed (17/17)	—
Test Sharpe (frozen)		0.496

K.2 Root Cause Analysis

Even with our tuned configuration, HyperAgent's failure modes illuminate why AEL's design choices are necessary.

First, **code generation quality collapses** beyond the initial improvement. While providing real tool-output schemas enables generation 0 to produce working code (train Sharpe improves from -0.974 to -0.44), subsequent generations overwhelmingly fail validation (85% failure rate in the tuned version). The LLM generates increasingly complex code that includes blocked imports or access patterns incompatible with the sandbox. AEL avoids this by constraining code evolution to modular components with well-typed interfaces.

Second, **batch evaluation causes overfitting**. The strategies that pass validation and improve training Sharpe often overfit to the training regime: seed 789 achieves improved training performance but collapses to -0.32 at test time. AEL's online per-episode learning avoids this: the bandit continuously adapts based on recent feedback rather than optimizing a single batch metric.

Third, **HyperAgent lacks diagnostic capability**. When the allocation function fails, the meta-agent knows only the aggregate Sharpe but not which signals were misleading or which allocation decisions were wrong. AEL's reflection system diagnoses failure patterns and targets memory improvements, enabling targeted adaptation rather than wholesale code rewriting.

K.3 Structural Comparison

Table 16 summarizes the structural differences. The key insight is that modular evolution with credit assignment preserves *locality*: changes to one component do not destroy progress in others.

This is analogous to how biological evolution operates through modular gene regulation rather than genome-wide random rewriting. Notably, the improvements HyperAgent would need to succeed (sliding-window evaluation, modular code generation, per-component signals) would make it architecturally similar to AEL, suggesting that AEL's design reflects fundamental requirements for self-improving agents in complex sequential tasks.

L Dataset Details

M Tool Descriptions

All methods share the same 12-tool finance registry for fair comparison, so performance differences cannot be attributed to tool access alone. The registry is intentionally heterogeneous: some tools expose *raw state* from the cached market data, others compute *derived signals*, and a final layer produces *decision-oriented summaries*. This separation is important for AEL because different planners may rely on different levels of abstraction, while the credit-assignment mechanism must be able to inspect both low-level evidence and high-level recommendations.

The tool set is also deliberately redundant in a useful way. Price-based tools capture short-horizon market structure, fundamentals and DCF provide slower valuation anchors, analyst/options/earnings tools expose event-driven and sentiment information, and correlation/risk tools help the planner reason about portfolio-level diversification rather than single-ticker alpha alone. Table 19 summarizes the role of each tool in the experiments.

Two design choices are worth noting. First, the registry includes both primitive and aggregated tools rather than forcing a single abstraction level; this gives the planner freedom to use direct evidence when needed and high-level summaries when time is limited. Second, several tools overlap on purpose. For example, momentum, technicals, analyst sentiment, and composite scoring may all point in the same direction during a strong trend, but they diverge during regime shifts; those disagreements are exactly the kind of cross-module evidence that makes credit assignment informative in our setting.

N Planner and Memory Policy Families

For the main AEL benchmark configuration, the planner pool is initialized with six built-in planners and the memory-policy registry is initialized with five default retrieval policies before any learned variants are added. These are the concrete families from which the meta-controller selects during training. The simpler incremental ablations intentionally restrict this space, often to a single sequential planner or a reduced memory setup, but the full benchmark uses the richer families summarized below.

These planners are intentionally diverse. Some are exhaustive (`sequential`), some are decompositional (`decompose`, `cot_reasoning`), and some are selective (`adaptive`, `reflexion`, `hypothesis_test`). The contextual planner bandit does not assume any one reasoning style is globally best; instead, it learns which planning style works better under which market context. In addition, procedural and semantic memory can later modify planner behavior by appending learned strategy hints to planner prompts.

The key design choice is that memory policies control *how* experience is exposed to the planner, not just *whether* memory exists. The policy determines which tiers are visible, how many memories are retrieved, and whether the planner sees raw cases, compact abstractions, or a larger but noisier experience bundle. This is why the memory-policy bandit is necessary: the best retrieval strategy depends on the task context and on the maturity of the memory store itself.

Algorithm 2 Inference workflow of AEL

1: Freeze module pools, bandit posteriors, and memory writes.
2: **for** each test episode e_t **do**
3: Extract task features ϕ_t.
4: Select (p_t, z_t, m_t) using the frozen meta-controller.
5: Retrieve read-only memory under policy m_t.
6: Execute planner p_t with tools z_t to produce $\hat{y}_t$.
7: Return prediction (no updates, reflection, or evolution).
8: **end for**

O Inference Workflow

P Detailed Analysis

P.1 LLM Credit Assignment Example

Below is a representative LLM credit assignment output from episode 85 (seed 42, bearish regime). The LLM receives each module's output alongside the ground-truth bar return, and assigns credit scores in $[-1, +1]$.

Input context: Bar 85 (2025-02-10 14:30).
Actual portfolio return: -0.31%.

Table 16: Structural comparison: why modular evolution with credit beats unconstrained code rewriting.

Dimension	HyperAgent	AEL
Modification scope	Entire function	Individual modules
Learning signal	1 Sharpe / generation	1 reward / episode
Diagnostic capability	None (aggregate)	Reflection-based
Evolution cost	$\sim$20 LLM calls	$\sim$400 LLM calls
Failure mode	Overfits or stuck	Gradual improvement
Test result (5 seeds)	0.46$\pm$0.41	**2.13$\pm$0.47**

Table 17: Full D-full benchmark statistics. Training includes diverse regimes (bull, bear, flat); the test set contains a bear-to-bull transition.

Property	Value
Task	Portfolio weight allocation (10 tickers + cash)
Tickers	10 sector-diverse equities
Sectors	7 GICS sectors (Tech, Healthcare, Finance, ...)
Frequency	1-hour bars (4 per trading day)
Train period	Jan 6 – Feb 21, 2025 (7 weeks, 140 bars)
Train regimes	3 bull + 2 bear + 2 flat weeks
Validation	Feb 21 – Mar 7, 2025 (2 weeks, 40 bars)
Test period	Mar 7 – Mar 21, 2025 (2 weeks, 28 bars)
Test regimes	Bear (W11) $\rightarrow$ Bull (W12) transition
Total episodes	208 bars
Action space	Weight vector $w \in \Delta^{10}$ (simplex)
Reward	Per-bar portfolio return

Table 18: D-full: 10 sector-diverse tickers spanning 7 GICS sectors.

Ticker	Sector	Market Cap	Train/Val/Test
AAPL	Technology	Mega	60/20/20
NVDA	Technology	Mega	60/20/20
JNJ	Healthcare	Mega	60/20/20
UNH	Healthcare	Mega	60/20/20
JPM	Finance	Mega	60/20/20
GS	Finance	Large	60/20/20
XOM	Energy	Mega	60/20/20
PG	Consumer	Mega	60/20/20
CAT	Industrial	Large	60/20/20
NEE	Utilities	Large	60/20/20

Planner selected `adaptive` strategy $\rightarrow$ overweighted tech (NVDA 18%, AAPL 15%).
Tools: `compute_momentum` signaled bullish for NVDA (+0.8); `compute_quant_risk` flagged elevated VIX (score 7/10); `score_composite_signal` returned HOLD.
Memory: Retrieved semantic memory: "Tech rally stalls when VIX exceeds 20" (relevance 0.72).

LLM credit output:
`planner: -0.6` *"The adaptive planner overweighted tech despite conflicting risk signals. It should have reduced concentration when VIX was elevated."*
`tools: +0.3` *"Momentum correctly identified the short-term trend, but quant_risk provided a valid warning that was ignored by the planner. Net positive for tools."*
`memory: +0.4` *"The retrieved memory about VIX and tech was highly relevant and should have been weighted more heavily. Memory retrieval was helpful; the failure was in how the planner used it."*

This illustrates how LLM credit can distinguish "the tool gave a correct warning that was ignored" from "the tool gave a wrong signal," a distinction that Shapley-based methods cannot make because they treat modules as black boxes.

P.2 Code Evolution Examples

The following are real artifacts produced by AEL's code evolution mechanism during training.

LLM-generated planner class (episode 120, seed 42). After reflection diagnosed that the `sequential` planner was too slow to react to intraday reversals, the LLM generated a new `MomentumReversalPlanner`:

```
class MomentumReversalPlanner(BasePlanner):
"""Reduce position when momentum
reverses intraday."""
def plan(self, context):
signals = context["tool_outputs"]
momentum = {t:
s["compute_momentum"]["trend_score"]
for t, s in signals.items()}
prev_momentum =
context.get("prev_momentum", {})
weights = {}
for ticker in context["tickers"]:
curr = momentum.get(ticker, 0)
prev = prev_momentum.get(ticker, 0)
if curr * prev < 0: # reversal
weights[ticker] = 0.05 # minimal
position
else:
weights[ticker] = max(0.02, 0.1 *
abs(curr))
return self.normalize(weights)
```

LLM-generated memory retrieval policy (episode 95, seed 123). After observing that default retrieval returned too many irrelevant bull-market memories during a bear regime, the LLM designed a regime-filtered retrieval policy:

```
class RegimeFilteredRetrieval(BaseRetrievalPolicy):
"""Filter memories by current regime
```

Table 19: Finance tools used in all experiments. The appendix view here emphasizes not only what each tool returns, but why that signal is useful for sequential portfolio allocation.

Tool	Main output	Why it matters
Data retrieval tools		
`get_price_history`	Recent OHLCV bars, latest close, highs/lows, trading volume	Anchors every price-based decision. It gives the planner direct access to recent regime, trend, and liquidity information, and it is the upstream dependency for several derived tools.
`get_fundamentals`	Financial statements, profitability, leverage, growth, and valuation ratios	Provides a slower-moving view of firm quality and balance-sheet strength. This is important because short-term price moves alone can be noisy, while fundamentals offer a medium-horizon anchor for capital allocation.
`get_analyst_data`	Target prices, consensus recommendations, upgrade/downgrade history	Adds an external sentiment and expectations channel. Analyst revisions can signal changing market narratives that are not yet obvious from raw prices.
`get_options_data`	Implied volatility, put/call ratios, open interest summaries	Exposes forward-looking positioning and hedging demand. This is especially useful for distinguishing bullish price action from fragile, high-volatility moves.
`get_earnings_data`	Quarterly earnings, revenue, and earnings-calendar metadata	Captures event risk and recent fundamental surprises. Around earnings windows, the planner needs to know whether a signal is driven by a durable trend or a transient post-event reaction.
Computation tools		
`compute_technicals`	RSI, MACD, Bollinger bands, moving averages, support/resistance, technical score	Converts raw prices into structured timing signals. These indicators help the planner reason about overbought/oversold conditions, trend continuation, and whether current prices are stretched relative to local history.
`compute_quant_risk`	Realized volatility, VaR/CVaR, Sharpe, Sortino, max drawdown, beta/alpha	Makes downside risk explicit rather than implicit. In portfolio allocation, avoiding bad concentration and tail exposure is as important as finding upside, so this tool supports risk-aware sizing decisions.
`compute_momentum`	Multi-horizon returns, trend slope, trend strength, volume trend	Measures continuation across several horizons instead of relying on a single look-back. This matters because different assets express momentum at different speeds, and volume confirmation helps separate genuine trends from weak drift.
`compute_correlations`	Cross-ticker correlation matrix and rolling correlation to target ticker	Moves the planner from single-name prediction to portfolio construction. High correlation means apparently strong single-stock signals may be redundant once existing exposures are considered.
Analysis tools		
`run_dcf_model`	Bull/base/bear DCF scenarios or simplified implied-upside valuation signal	Supplies an intrinsic-value estimate that can disagree with recent price action. This is useful for distinguishing momentum-driven trades from opportunities where valuation support exists.
`score_risk`	Overall 1–10 risk rating plus valuation/financial/growth/macro/technical sub-scores	Compresses several risk dimensions into a planner-friendly summary. This makes it easier to compare heterogeneous tickers and avoid allocations that are attractive on return but unacceptable on fragility.
`score_composite_signal`	Weighted BUY/SELL/HOLD style summary using technical, momentum, valuation, analyst, options, and risk inputs	Acts as a high-level synthesis layer. It is useful when the planner wants a compact recommendation, but it is also diagnostically important because the credit module can inspect whether the fused signal helped or obscured the true decision.

```
    before scoring."""
    def retrieve(self, query, memories,
    k=5):
    regime = query.get("current_regime",
    "unknown")
    filtered = [m for m in memories
    if m.get("regime") == regime
    or m.get("tier") == "procedural"]
    if len(filtered) < k:
    filtered = memories # fallback
    scored = self.score_by_relevance(query,
    filtered)
    return scored[:k]
```

Both artifacts were validated via AST parsing and accepted into the module pool by LinUCB, which subsequently selected them when their context features matched.

Q Baseline Adaptation Details

All baselines use the same LLM backbone (Claude Haiku 4.5), tool set (12 financial tools), and portfolio allocation interface. Below we describe what was kept from each original method and what was adapted.

Reflexion (Shinn et al., 2023). *Kept:* The core mechanism of accumulating verbal self-critiques after each episode. Reflections are prepended to future prompts as a growing context window. *Adapted:* Applied to portfolio allocation with a financial-specific reflection prompt ("Write a 1-sentence reflection on what signals to trust or ignore"). Maximum 20 reflections retained (FIFO eviction). No structured memory tiers; only flat string accumulation.

ExpeL (Zhao et al., 2024). *Kept:* The experience extraction mechanism where the LLM distills episodes into reusable "lessons" stored in a flat lesson store, retrieved by keyword similarity. *Adapted:* Lessons are keyed by ticker and sector for retrieval (same-ticker: +2.0, same-sector: +1.0). Maximum 100 lessons. Extraction prompt asks for generalized rules prefixed with "RULE:". No bandit-based policy selection.

FactorMiner (Wang et al., 2026b). *Kept:* Dual-tier memory with skill extraction from successful tool combinations and experience memory with outcome tracking. *Adapted:* Skills are named by sorted tool sequences from correct predictions ($\geq$2

Table 20: Built-in planner families used by AEL in the main benchmark configuration. Dynamic planners generated by the slow-timescale evolution loop are added on top of this initial pool.

Planner	Core principle	Role in the benchmark
`sequential`	Run all available tools in a fixed order, then synthesize once.	Serves as the most stable and exhaustive baseline planner. It maximizes coverage and minimizes strategic assumptions, but can be expensive and prone to information overload.
`decompose`	Break the task into valuation, momentum, sentiment, and risk sub-problems, then synthesize sub-results.	Encourages structured analysis and lets the agent reason about different evidence types separately before combining them into a portfolio decision.
`adaptive`	Start with a cheap quick-look tool set, then invoke deeper tools only if the initial signal is ambiguous.	Provides an efficiency-oriented planner that trades off speed and depth, which is useful when some episodes are easy while others require broader evidence.
`cot_reasoning`	Analyze trend, valuation, sentiment, and risk sequentially with explicit intermediate synthesis.	Forces a chain-of-thought-style evidence path rather than a flat aggregation of all tool outputs, which can help when conflicting signals need to be resolved step by step.
`reflexion`	Make a quick prediction, self-check confidence, and gather more evidence only if the initial judgment is weak.	Adds an intra-episode self-correction behavior: the planner first tests whether existing evidence is sufficient and only expands the search when confidence is low.
`hypothesis_test`	Form bull and bear hypotheses, gather targeted evidence for each, then weigh the two cases.	Makes the planner explicitly compare competing market narratives instead of only aggregating signals, which is useful in reversal or mixed-regime episodes.

Table 21: Initial memory-policy families used by AEL in the main benchmark configuration. New retrieval policies may be added later by the slow-timescale evolution loop.

Policy	Enabled tiers	Format	Retrieval principle
`none`	none	`none`	Disable memory entirely. This is the no-retrieval option and is important because some episodes are better solved from current market evidence alone.
`recent_window`	episodic	`sliding_window`	Retrieve recent episodic memories and keep only a small first-plus-last window. This preserves a few anchors and a few recent cases without flooding the planner with raw logs.
`full_detailed`	episodic, semantic, procedural	`full`	Return all retrieved memories from all tiers verbatim. This is the highest-information policy, useful when the agent benefits from both concrete cases and abstract rules.
`compressed`	semantic, procedural	`ranked_truncate`	Focus on abstracted knowledge, rank retrieved memories by relevance, and truncate to a token budget. This is the default "high signal, low clutter" option for using distilled experience.
`aggressive_learner`	episodic, semantic, procedural	`ranked_truncate`	Retrieve more memories with a larger token budget and a more permissive learning setup. This policy is useful when the system is still exploring and wants to exploit a broader experience base.

tools). Success rates use exponential moving average. Maximum 15 skills and 200 experiences. No joint evolution or credit assignment across modules.

Meta-Reflexion (Wu et al., 2025). *Kept:* Rule distillation from accumulated reflections, with admissibility checking that prunes contradicted or low-success rules. *Adapted:* Distillation runs every 5 episodes, extracting "RULE:" lines from the 10 most recent reflections. Rules with <0.3 success rate and ≥ 5 applications are pruned. Maximum 10 active rules. No tool or planner evolution.

EvoTool (Yang et al., 2026). *Kept:* Population-based evolutionary optimization of tool-selection policies with blame-aware mutation. Fitness-proportional selection across a population of 5 policies. *Adapted:* Blame attribution uses signal-level analysis (e.g., if `compute_momentum` signaled bullish but the actual direction was bearish, that tool is blamed). Mutations remove blamed tools, add random tools (50% chance), or swap tools (30% chance). Minimum 3 tools per policy. No memory system or planner evolution.

R Transaction Cost Sensitivity

A practical concern for portfolio allocation systems is whether the reported Sharpe ratios survive realistic transaction costs. Since the D-full benchmark uses hourly bars (4 per trading day), frequent rebalancing can generate substantial turnover. To assess this, we retroactively apply proportional transaction costs to the recorded portfolio weight changes at each bar. Specifically, for a cost level of c basis points per unit of turnover, the cost-adjusted return at bar t is $r_t^{\text{adj}} = r_t - c \cdot \sum_i |w_{i,t} - w_{i,t-1}|$, where $w_{i,t}$ is the portfolio weight of ticker i at bar t.

Table 22 reports cost-adjusted Sharpe ratios at four cost levels spanning the range from zero-cost (0 bp, the main benchmark setting) to institutional-level costs (20 bp, typical for large-cap equity rebalancing with market orders).

Several observations are worth noting. First, AEL remains the top-performing method at all cost

Table 22: Cost-adjusted Sharpe ratio at varying transaction cost levels (5-seed means). Costs are applied retroactively based on recorded turnover at each bar.

Method	0 bp	5 bp	10 bp	20 bp
AEL	2.13	∼1.9	∼1.7	∼1.3
momentum_weighted	1.44	∼1.3	∼1.2	∼1.0
EvoTool	1.37	∼1.2	∼1.0	∼0.7
Stateless	1.35	∼1.2	∼1.0	∼0.7

levels tested, maintaining a Sharpe above 1.3 even at 20 bp. Second, the cost degradation is moderate (∼0.8 Sharpe from 0 to 20 bp) because AEL's reflection mechanism tends to produce stable allocation strategies that avoid excessive turnover; the agent learns through reflection that frequent large weight shifts are penalized by the market. Third, the momentum-weighted baseline is relatively cost-resilient (Sharpe 1.0 at 20 bp) because its allocation weights change smoothly by construction, while EvoTool's evolutionary mutations can produce abrupt policy changes that incur higher turnover costs. The cost estimates at 5, 10, and 20 bp are based on average turnover statistics from logged portfolio histories and should be interpreted as approximate; the `cost_adjusted_sharpe` function in the codebase enables exact computation from individual run logs.

S Running Case — Full Trace

The main text (Section 3.5) presents an abridged view of seed 42's episodes 83–96. The full trace, including per-episode reward sequences, posterior parameter updates, and the exact reflection prompts and responses, is reproducible from the logs under `logs/pf_eael_incremental_d_full_s42_13ba204/`. The summarised dynamics are: episodes 78–82 alternate between `aggressive_learner` (mean reward 0.41) and `recent_window` (mean reward 0.49); episode 83 triggers an under-threshold loss that initiates the slow-window reflection; episode 84's reflection introduces `regime_filtered_retrieval` as a sixth bandit arm; episodes 85–95 show TS sampling the new arm with rising probability (from 0.18 to 0.34); the new arm's posterior mean reaches 0.55 by episode 96, surpassing the original five-policy maximum.

T Reflection Faithfulness — Judge Transcripts

The LLM-as-judge evaluation reported in Table 8 (R2) uses Claude Opus 4.7 as a separate judge model. We provide raw judge transcripts in the supplementary materials. The judge prompt template, anchor examples, and scoring rubric (5 = "all factual claims verifiable from the slow-window summary; causal claim testable; prescription actionable", 0 = "contradicts the data, or vacuous") are released alongside the code. Inter-window judge variance was $\sigma \in [0.4, 0.7]$ across the three axes, computed by sampling two independent judge passes per reflection on a 12-reflection subsample.

U Synthetic Cross-Domain Benchmark Details

The synthetic bandit used in Section 4.5 has 10 initial arms and one hidden 11th arm injectable only via reflection. Per-regime Bernoulli probabilities are:

Reflection trigger. The TS+reflection algorithm runs a slow-window check every 13 episodes. If the trailing 25-episode mean reward is below 0.42 and the new arm has not yet been added, reflection injects the hidden 11th arm as a new bandit arm with a uninformative Beta$(1, 1)$ posterior. The arm is then sampled by TS like any other.

Why the trigger threshold is 0.42. Pure Thompson Sampling on this benchmark converges to a steady-state mean of approximately 0.40 when no regime shift has degraded its current best arm. Setting the threshold at 0.42 ensures reflection fires only when the bandit has slipped well below its expected operating point, mimicking the AEL paper's behaviour of triggering reflection only when the current memory pool is underperforming.

Seeds and reproducibility. 5 seeds (42, 123, 456, 789, 1024), 208 episodes each. Code released as `synthetic_crossdomain.py` alongside the paper. Per-seed final reward streams reproducible by running the script directly; no LLM calls required.

What we did not run. Beyond the support-ticket routing LLM-agent stream in Appendix V, we deliberately do not present results on larger real-world agent benchmarks (e.g., SWE-bench, WebArena, HotpotQA) because running 5 seeds × 208 episodes with matched baselines would cost roughly an order of magnitude more LLM calls than the financial benchmark. We commit to releasing such extensions in follow-up work; the current paper's cross-domain scope is mechanism validation rather than broad task coverage.

Table 23: Per-arm Bernoulli reward probabilities by regime. Each regime makes a different initial arm best at 0.65; the hidden arm pays 0.70 in regime 3 only.

Regime	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10 (hidden)
0 (ep 0–50)	0.30	0.35	**0.65**	0.40	0.45	0.30	0.25	0.30	0.35	0.40	0.30 (locked)
1 (ep 50–100)	0.25	0.30	0.35	0.30	0.35	0.40	0.30	**0.65**	0.35	0.30	0.30 (locked)
2 (ep 100–150)	**0.65**	0.30	0.35	0.30	0.30	0.35	0.30	0.30	0.40	0.35	0.30 (locked)
3 (ep 150–208)	0.30	0.35	0.30	0.30	0.35	**0.65**	0.30	0.30	0.35	0.40	**0.70**

V Support-Ticket Routing Stream LLM-Agent Details

The support-ticket routing stream contains 5 generated JSONL files with 1,200 support tickets each (seeds 42, 123, 456, 789, 1024). The real LLM-agent table uses a stratified 160-episode subset per seed (40 per regime), for 800 episodes per method. All calls use Gemini 3.1 Flash-Lite via LLM API calls, batch size 8, temperature 0, and default error policy `raise`. The run made 891 LLM calls, with 0 errors and 0 fallbacks. The main-text table in Table 2b excludes the diagnostic `fixed_compressed` ablation: it is a useful retrieval-policy probe, but not a prior-method baseline.

Baseline adaptations. Reflexion stores recent success/failure self-critiques as growing context. ExpeL stores lessons retrieved by endpoint, regime, and token overlap. Meta-Policy Reflexion distills failure rules and prunes low-success rules after repeated applications. EvoTool evolves a small population of retrieval-tool policies by recent fitness. FactorMiner stores endpoint/regime skills and experience memories. HyperAgent is not included because its portfolio baseline rewrites an allocation function; the support-ticket routing task compares online memory/reflection mechanisms rather than recursive code generation.

Support-ticket routing hyperparameter sensitivity. To avoid spending additional API calls on a grid search, we run the hyperparameter sweep on the deterministic full stream (1,200 episodes per seed) using the same reward and reflection-injection logic. The artifact is released as `cli_support_stream_hp_sensitivity.json`. Table 24 shows that settings which inject the route-aware policy at episode 920 recover the full R3 gain, while slower cadence or a threshold that delays injection to episode 960 loses roughly 8.9pp R3 accuracy. This supports the main interpretation: the mechanism is not fragile to small parameter changes, but it is sensitive to whether reflection happens early enough in the regime where the new policy matters.

W Procedural Memory Growth Curve

Table 25 shows the number of active procedural rules across training for each of the 5 seeds. The promotion rule (corroboration in ≥ 3 slow windows with consistent direction) keeps the rule pool small while the pruning rule (rolling-10 success < 0.4) discards rules that lose relevance as regimes shift. Median end-of-training counts are 7–11 active rules per seed; the maximum observed at any training step is 19 (seed 456, episode 142), after which two redundant momentum/reversal rules are pruned by episode 158.

Table 24: Support-ticket routing reflection hyperparameter sensitivity on the deterministic full stream (5 seeds, no LLM calls). The main insight is timing: delayed injection hurts the route-aware R3 regime.

Setting	Injection episode	Overall	R3
Default: every 40, window 120, threshold .58	920	0.700 ± 0.011	0.863
Faster cadence: every 20	900–920	0.703 ± 0.009	0.875
Slower cadence: every 80	960	0.677 ± 0.011	0.774
Longer window: 200	920–960	0.692 ± 0.013	0.831
Lower threshold: .50	960	0.677 ± 0.011	0.774
Higher threshold: .66	920	0.700 ± 0.011	0.863

Table 25: Active procedural-rule count over training (5 seeds). Counts are computed every 20 episodes; columns are training-episode checkpoints.

Seed	ep 20	ep 40	ep 60	ep 80	ep 100	ep 120	ep 140	ep 166
42	2	4	5	6	7	9	9	8
123	1	3	4	6	7	8	9	9
456	3	6	9	12	14	17	19	11
789	2	3	5	6	7	8	9	7
1024	2	4	6	8	9	10	10	9
Mean	2.0	4.0	5.8	7.6	8.8	10.4	11.2	8.8