STREAMING^2: ACTING ON EVIDENCE STREAMS, EVOLVING THROUGH DISASTER EVENT STREAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Disaster-response agents revise their harnesses to improve later decisions by adapting programs and configurations governing evidence and tool use. These updates use feedback from comparing forecasts with later observations. However, forecasts may share inputs or be checked against the same observations. Treating their feedback as independent can overstate support for retaining updates. Existing benchmarks do not trace how these dependencies affect updates and later performance. We introduce Streaming^2, a benchmark annotating forecast lineage and measuring how updates affect later tasks. Its 1,000 events contain 10,000 action-selection and spatial-analysis tasks, with evidence arriving within events and harnesses evolving across them. Validation experiments show that a baseline agent makes fewer harmful updates when given these annotations. However, some updates still improve performance on past tasks but not on future events. These findings motivate FLARE, which compares content along forecast lineage to generate complete updates and predict their future gains. It learns from measured performance differences between updated and unchanged harnesses on the same later training events. At test time, learned parameters remain frozen while predicted gains guide retention. On Streaming^2-Bench, FLARE raises the overall task score by 21.0% relative to the fixed harness and 4.4% relative to MemEvolve, the strongest self-evolving baseline.

1 INTRODUCTION

Disaster-response agents make decisions as evidence arrives, and experience from completed events can guide their responses to later ones. Within each event, agents select actions or produce spatial assessments using the forecasts, satellite imagery, and ground observations available at that decision point (Wang et al., 2026; Varambally et al., 2026). Comparing these forecasts with later observations provides feedback on the predictions that informed those decisions. After the event, an agent can use this feedback to revise its harness—the programs and configurations that govern evidence and tool use (Shao et al., 2026). These revisions persist across events, changing how the agent handles later tasks while its underlying foundation model remains fixed. Figure 1 illustrates decisions within events and harness evolution across events.

However, more feedback records do not necessarily provide more independent evidence for a persistent harness change. Forecast versions may share inputs or be checked against the same observations, so their feedback can reflect overlapping evidence. Yet revised forecasts can still contain new information relevant to the agent's decisions. An agent updating its harness may overcount shared evidence or dismiss useful revisions, retaining rules that work poorly in later events. This raises our first question: How do relationships among forecasts and their feedback affect persistent harness updates and later task performance?

Answering this question is difficult with existing forecast and agent benchmarks. WeatherBench 2 measures forecast quality against reference weather data (Rasp et al., 2024), while DORA and GeoDisaster evaluate geospatial reasoning, tool use, and disaster-task performance (Wang et al., 2026; Hasan et al., 2026). These benchmarks do not follow feedback-driven harness changes across events. LifelongAgentBench extends evaluation to experience reuse across sequential tasks linked by shared skills (Zheng et al., 2025). These task relationships do not capture shared evidence among

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

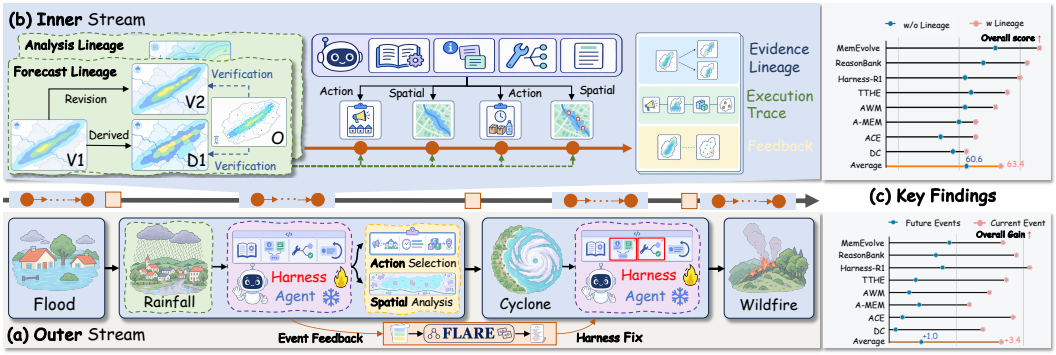

Figure 1: Streaming^2 overview. Figure 1(a) shows feedback updating the harness that controls evidence and tool use across events. Figure 1(b) follows decisions as evidence arrives, linking revised and derived forecasts to shared verification observations. Figure 1(c) shows complete-stream scores with and without lineage annotations above. Below, each method's retained edits are compared on original-task reruns and later events. Gains are mean score differences between updated and unchanged harnesses on the 0–100 scale. Each Average row weights the eight methods equally.

forecast-feedback records. We therefore need to trace which related feedback records support a retained update and measure how that update changes performance on later events.

To support this evaluation, we introduce Streaming^2, a benchmark for agent evolution with 1,000 disaster events and 10,000 task instances. It comprises Streaming^2-Evo for training and validation and Streaming^2-Bench for held-out evaluation. Within each event, agents select actions or produce spatial assessments using the evidence available at each decision point. Across events, agents can revise their harnesses using released feedback and retain changes for subsequent tasks. To identify related feedback, we annotate forecast lineage: how forecasts share inputs, revise or derive from earlier products, and are checked against common observations. Execution traces record the feedback used to propose each harness update and whether the agent retains it. To measure an update's later effect, we compare task performance under updated and unchanged harnesses on the same subsequent events. Within the training split, these differences provide gain labels for complete updates, with each candidate's changes evaluated together.

Diagnostics on Streaming^2-Evo's validation split reveal two difficulties in retaining harness updates that improve later task performance. Given the same forecast content and released feedback, a baseline agent more often retains updates that worsen later performance when relationship annotations are withheld. Even with these annotations, some retained updates improve performance when the original tasks are rerun but not on later events. These findings motivate our second question: How can an agent use available experience and forecast relationships to assess a complete update's future gain over making no change?

To address this question, we propose FLARE, a framework using **F**orecast **L**ineage for **A**dvantage-guided **R**untime Harness **E**volution. To account for related feedback, its Lineage-Residual Encoder combines relationships and content differences among forecasts with recorded evidence use and released feedback. Using this experience representation and the current harness, the Lineage-Conditioned Evolver proposes complete candidate updates. The Edit-Gain Predictor uses the same experience to estimate each complete update's future gain over leaving the harness unchanged. Both generation and gain prediction are trained with measured gains on later Streaming^2-Evo training events. At runtime, FLARE uses predicted gains to retain an update or keep the current harness. On held-out Streaming^2-Bench events, FLARE improves action selection and spatial analysis over our two controls, Source-aware Reflection and the Text-conditioned Future-Gain Editor.

Our contributions are threefold:

- We introduce **Streaming^2**, a reusable benchmark that connects decisions as evidence arrives within disaster events with persistent harness evolution across events. Its relationship annotations identify related forecast feedback, and gains from paired runs support learning and evaluating complete harness updates.

- We propose **FLARE**, which uses forecast content and lineage to generate and evaluate complete harness updates. FLARE learns from measured gains on later training events and uses predicted gains to retain an update or leave the harness unchanged.
- Validation diagnostics show that a baseline agent more often retains updates with negative later gains when relationship annotations are withheld. Even with these annotations, improvements on original tasks need not transfer to later events. On Streaming^2-Bench, FLARE raises the overall task score by 21.0% relative to the fixed harness and 4.4% relative to MemEvolve, the strongest self-evolving baseline.

2 Related Work

Forecasting and Agent Benchmarks. Existing benchmarks evaluate forecasts, agent tasks, or experience reuse. WeatherBench 2 measures forecast quality (Rasp et al., 2024), while Zephyrus, DORA, and GeoDisaster evaluate weather or geospatial agents (Varambally et al., 2026; Wang et al., 2026; Hasan et al., 2026). LifelongAgentBench studies experience reuse across tasks (Zheng et al., 2025), and FutureSim tests prediction revision as information arrives (Goel et al., 2026). Their reported protocols do not trace how related forecast feedback shapes persistent harness edits and task performance on later disaster events. Streaming^2 enables this joint evaluation.

Experience-Driven Agent Evolution. Agent evolution methods differ in what they retain and how they assess updates. Reflexion, ACE, and GeoEvolver retain reflections, playbooks, or procedural memories, with ACE and GeoEvolver also consolidating experience (Shinn et al., 2023; Zhang et al., 2026; Dai et al., 2026). Harness-R1 learns executable edits from gains on the same task batch (Shao et al., 2026), while Milkyway revises forecasting guidance before a revisited question's outcome is known (Wei et al., 2026). We instead ask whether a complete edit informed by related forecast feedback should persist into later events.

Source Dependence and Edit Evaluation. Source dependence and edit evaluation address complementary parts of this decision. Truth discovery accounts for source copying (Dong et al., 2009), while STAMP traces evidence to the search actions that exposed it (Xu et al., 2026). SkillMaster measures skill-edit gains on common probes (Yang et al., 2026), and CHILL-Harness compares workflows continued from the same task state (Fu et al., 2026). FLARE combines verified forecast relationships with recorded evidence use and feedback available at edit time. It predicts each complete edit's gain on later events over leaving the harness unchanged.

3 Streaming^2: A Benchmark for Agent Evolution in Disaster Response

Streaming^2 evaluates decisions under incoming evidence within events and harness evolution across events. It organizes 10,000 Action Selection and Spatial Analysis tasks into sequences within 1,000 events, shown in Figure 2. Available evidence averages 24 forecast records and six satellite scenes per task at its decision point, alongside ground observations. Assets can recur across tasks, so these averages describe available evidence rather than distinct inventory per task.

To trace related feedback, verified lineage links connect 96,000 forecast records to input products and verification observations, covering 60% of 160,000 records. For learning which updates to retain, 8,000 training records pair complete edits with gains over unchanged harnesses on the same later training events. Streaming^2-Evo supports training and validation, while Streaming^2-Bench provides held-out testing.

3.1 Task Taxonomy

Each task type has six subtasks, with situations distinguished by what the evidence establishes and the task requires. Instances place these situations at decision points within events.

Action Selection has 4,000 instances across 20 basic decision situations. Its subtasks cover initial response selection, ongoing response adjustment, evidence requests, waiting and evidence substitution, response-plan selection, and resource allocation. Agents submit complete plans that satisfy

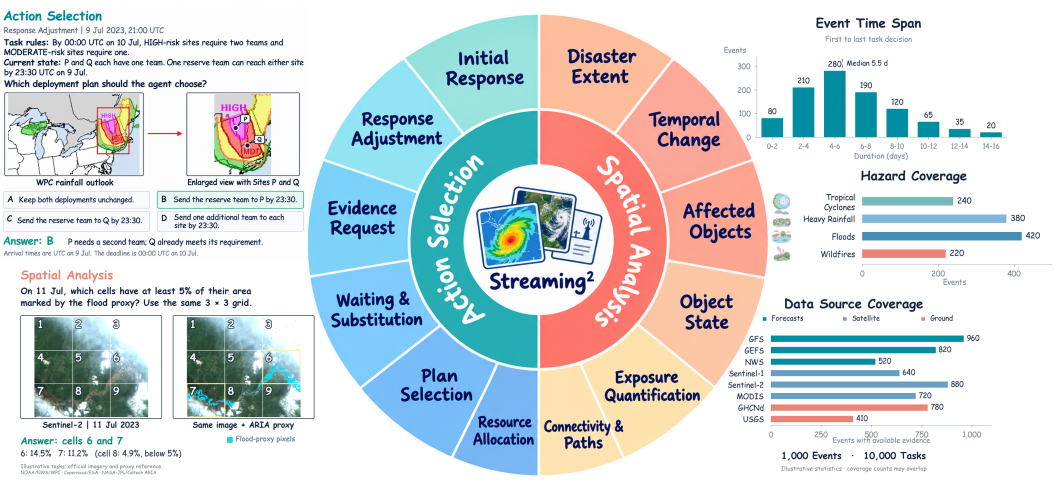

Figure 2: Streaming^2 tasks and event coverage. The wheel groups Action Selection and Spatial Analysis into six subtasks each. Left-hand examples show response-plan selection and flood-proxy cell selection. Right-hand plots show time from first to last task decision and event counts by hazard and source. Events can count toward multiple hazards and sources.

stated rules using current evidence. Additional requirements can cover temporary responses or transport and operational constraints on resource allocation.

Spatial Analysis has 6,000 instances across 15 situations. Disaster extent extraction maps hazard coverage, and temporal change analysis tracks it across observations. Affected-object identification selects objects by their spatial relation to hazards, while exposure quantification measures exposed populations, assets, or areas. Object-state assessment requires evidence of damage or functional status beyond spatial exposure. Connectivity and path analysis determines reachability, distances, and routes under stated access constraints.

3.2 Event Construction and Evidence Lineage

We construct each event from archived forecasts, satellite and ground observations, and geographic data for its region and period. Availability times place these assets on a timeline, from which each task receives only evidence accessible at its decision point.

Reference answers combine large-model assistance with final approval by human domain experts. For Action Selection, models propose acceptable actions and conditions from decision-time evidence and rules. Experts correct these proposals before approved conditions become automated checks for complete plans. Spatial Analysis references come from existing annotations, spatial overlays, temporal comparisons, and network analysis. Models help fill missing spatial labels from observations, and experts verify all spatial references against source evidence.

Harness updates use forecast-verification feedback whose lineage annotations identify related records. Each record compares a forecast version with observations of the same variable, region, and target period. Product metadata and documentation link forecasts to inputs, replaced versions, and source products. Together with the verification pairs, these links identify shared inputs or observations. Feedback becomes accessible only at its recorded release time. Forecast-verification feedback is distinct from private task scores and edit-gain labels.

3.3 Streaming Protocol and Evaluation

The streaming protocol links task execution within events to harness updates between events. Each event uses an unchanged harness and only evidence available at each decision. After its tasks finish and required feedback arrives, the agent may commit one complete update or retain its harness for the next event.

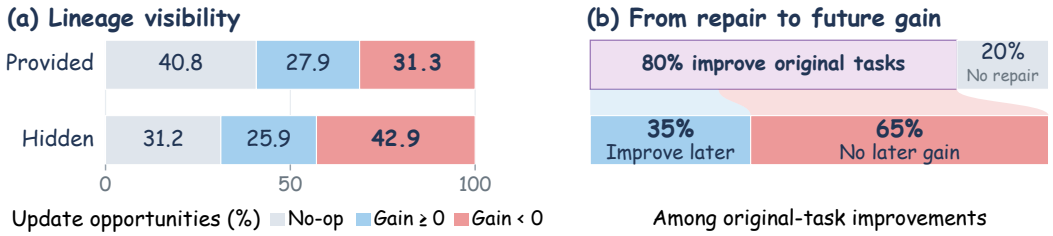

Figure 3: Baseline lineage diagnostics on Streaming^2-Evo validation events. Figure 3(a) divides all update opportunities into no change and retained edits with nonnegative or negative later gains. Figure 3(b) follows edits retained with annotations: 80% improve original-task reruns, but 65% of these do not improve later events. Gains average updated-minus-unchanged task scores. See Table 5.

Event scores balance Action Selection and Spatial Analysis. Action scores are binary, indicating whether complete plans satisfy reference conditions. Spatial scores compare results against references using output-specific measures. We average scores within each task type and weight the two means equally, yielding $S_e(H)$ for harness H on event e.

At update opportunity t, we measure the later effect of candidate edit δ by comparing the updated harness H_t^δ with the unchanged harness H_t. Both remain fixed during matched paired runs on the same K later events, yielding

$$\bar{\Delta}_t(\delta) = \frac{1}{K} \sum_{j=1}^{K} \left[\bar{S}_{e_j}(H_t^\delta) - \bar{S}_{e_j}(H_t) \right]. \tag{1}$$

Here, $\bar{S}_e(H)$ averages event scores over repeated runs, detailed in Appendix B. Training records pair each candidate and its proposal-time experience with this measured gain on later training events. Later outcomes supply labels only, never inputs to candidate generation or selection.

3.4 Quality Control, Data Splits, and Reuse

Quality control checks reference answers and evidence timelines before evaluation. Human domain experts resolve annotation disagreements against source evidence and task rules, excluding unresolved cases. We then freeze references and scoring rules. Availability times use documented records or explicitly marked reconstructions, detailed in Appendix A.

To limit leakage, events from the same real-world disaster stay in one split. Streaming^2-Evo contains 640 training and 160 validation events, and Streaming^2-Bench contains 200 test events. Learned components remain fixed during testing while harnesses evolve under the protocol in Section 3.3. Paired runs stay within each split, keeping training gain labels separate from validation and test outcomes. Table 10 summarizes resource composition.

Shared event streams and scoring interfaces support comparisons of evolution methods. Training records trace each gain to its source harness, complete edit, and paired runs.

4 Findings and Method

4.1 Benchmark Findings

On Streaming^2-Evo's validation split, we diagnose a baseline combining a Harness-R1-style edit generator with an LLM critic through paired evaluations (Section 3.3). We examine how lineage annotations affect which edits the agent retains and whether improvements on the original tasks carry over to later events.

Lineage annotations reduce harmful updates. We vary only the visibility of lineage annotations during edit generation and selection. With annotations, updates with negative later gains are retained at 31.3% of all update opportunities, compared with 42.9% without them in Figure 3a. These results motivate using forecast relationships alongside content when generating and evaluating persistent updates.

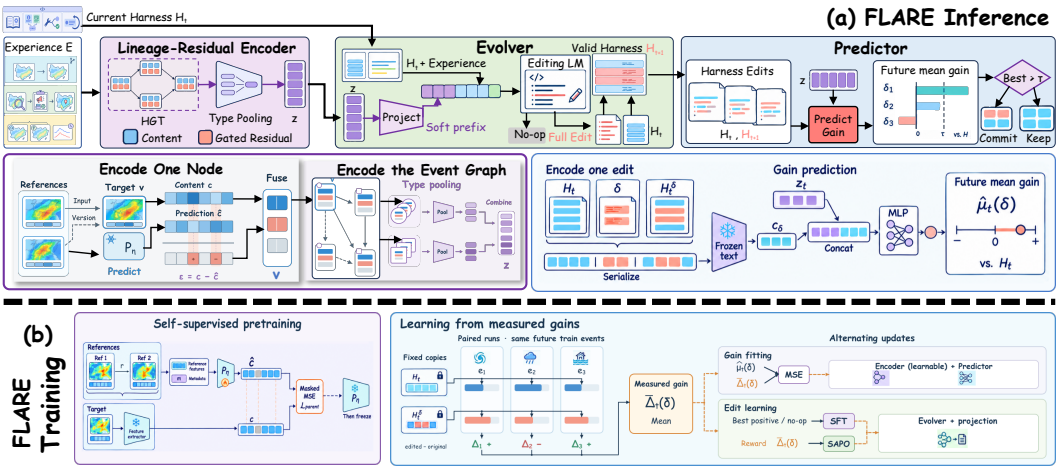

Figure 4: FLARE inference and training. Figure 4(a) encodes forecast content and lineage as z_t to generate complete edits and predict mean later score gains over the unchanged harness. Only the best valid edit with predicted gain above τ is retained. Figure 4(b) freezes pretrained content predictor P_η. Measured gains from paired later training runs supervise prediction and reward generation.

Updates can repair past tasks without improving later performance. For edits retained with lineage annotations, we compare updated and unchanged harnesses on reruns of the original tasks and on later events. Among the edits that improve the original tasks, 65% yield zero or negative average gains on later events in Figure 3b. These cases motivate estimating future gains from available experience before deciding whether to retain an update.

4.2 The FLARE Framework

FLARE uses forecast content and lineage to generate updates and predict their later gains, as Figure 4 shows. The Lineage-Residual Encoder represents experience through forecast relationships and content differences. The Lineage-Conditioned Evolver uses this representation and the current harness to propose complete updates. The Edit-Gain Predictor estimates their future gains over making no change, guiding retention.

At update opportunity t, FLARE uses the current harness H_t and only the experience then available, $\mathcal{I}_t = (G_t, \mathcal{T}_t, \mathcal{F}_t)$. Here, G_t contains available products and verified relationships, $\mathcal{T}_t$ records evidence use and actions, and $\mathcal{F}_t$ contains released feedback.

Lineage-Residual Encoder. The Encoder represents forecast relationships while preserving content differences relevant to updates. Graph G_t connects available products through verified input, derivation, and version links. Shared-verification links connect forecasts whose released feedback uses the same observation record and version. To capture content differences, we compare each product with a prediction from related products. Frozen modality-specific encoders or fixed numerical transformations extract node features c_v. We predict these features from verified upstream products and comparable earlier versions. Their features, metadata, and relation types form $\mathcal{R}_t(v)$, while target metadata m_v specifies type, time, and region. The frozen predictor P_η estimates c_v in the target type's fixed feature space, leaving residual ϵ_v:

$$\widehat{c}_v = P_\eta(m_v, \mathcal{R}_t(v)), \qquad \epsilon_v = c_v - \widehat{c}_v. \tag{2}$$

To guide edits, the Encoder combines these content differences with evidence use and released feedback. A residual measures unexplained content, not statistical independence or task value. A learned gate adjusts each residual's contribution using content, use records, and feedback. We retain original content because predictable information may still matter for tasks. Content, gated residuals, metadata, use records, and released feedback form node representations in a common space, as detailed in Appendix C. A Heterogeneous Graph Transformer (Hu et al., 2020) aggregates them along product-lineage and shared-verification links. Type-wise pooling yields z_t for edit generation and gain prediction.

Lineage-Conditioned Evolver. The Evolver uses a language model separate from the task executor to propose edits to H_t for later events. A trainable projection maps z_t to continuous prompt vectors, following G-Retriever (He et al., 2024). For concrete editing details, the model also receives H_t and text describing available evidence, relationships, execution history, and released feedback. From these inputs, the model generates an edit δ or `no-op`.

FLARE evaluates each edit as the complete harness that would execute. Edits target four fixed execution points, following Harness-R1 (Shao et al., 2026). A deterministic materializer applies all changes to H_t together, producing H_t^δ. Only candidates passing interface and permission checks reach the Gain Predictor; Appendix B specifies these checks.

Edit-Gain Predictor. The Gain Predictor estimates an edit's average later gain over leaving H_t unchanged. Changes within an edit may interact, and their effects depend on task conditions. We therefore evaluate the complete edit alongside proposal-time experience z_t. A frozen text encoder jointly embeds H_t, δ, and H_t^δ into c_δ. An MLP mean head predicts $\widehat{\mu}_t(\delta) = f_\omega([z_t; c_\delta])$, supervised by measured later-training-event gains $\bar{\Delta}_t(\delta)$ from Section 3.3.

Edit Selection. FLARE commits the valid edit with the highest predicted gain only if that gain exceeds a validation-selected threshold $\tau \geq 0$. Otherwise it keeps the current harness. Appendix B specifies candidate and tie rules. Predicted positive gains do not guarantee realized improvements.

4.3 Training and Inference

Training first establishes content comparison and edit evaluation, then teaches generation from measured gains. We pretrain P_η on product features and freeze it to stabilize residuals. Initial candidate edits are proposed from harness states accumulated by a fixed collector and receive labels from paired later-event runs. These gains jointly supervise the Encoder's learnable components and the Gain Predictor. With both fixed, the Evolver learns each state's best evaluated valid positive-gain edit, or `no-op` when none improves performance.

FLARE then collects its own states and candidates on further training events. Policy updates reward the Evolver with measured gains, alternating with Encoder and Predictor updates from accumulated labels. The Encoder and Predictor stay fixed within each policy-update block. After they are refreshed, supervised fine-tuning adapts generation to the new representations. New collection begins beyond all events already used for learning or model selection. Appendix C details this schedule. Later outcomes supply targets and rewards without entering the original proposals or retention decisions.

After training, we freeze learned parameters and select τ by mean validation event score. Because thresholds change later harnesses and experience, each is evaluated over a complete validation sequence from the same initial harness. During testing, parameters and τ stay fixed while released feedback guides harness updates. Private paired-run gains serve evaluation only.

5 Experiments

We assess FLARE's overall effectiveness by comparing task performance over complete event streams. We then examine the contributions of experience representation, candidate generation, and edit selection, and assess transfer to hazard types excluded from offline learning.

5.1 Experimental Setup

Models and baselines. We use a frozen Qwen3.8-27B executor and adapt a separate copy with LoRA (Hu et al., 2021) as the Evolver (Appendix D). We compare FLARE with a fixed harness, twelve adapted methods, and two study-specific controls (Table 1). Source-aware Reflection (SA-Reflection) organizes feedback by forecast lineage before proposing edits. Text-conditioned Future-Gain Editor (Text-Gain) uses text representations for generation and gain prediction, with the same edit-gain supervision as FLARE.

Table 1: Performance over complete Streaming^2-Bench event streams. Action measures complete-plan success percentages. Spatial and Overall use 0–100 scores. W/T/L gives event win/tie/loss percentages against the fixed harness. Bold marks score maxima, including ties.

	(a) Action Selection							(b) Spatial Analysis							Overall	
Method	Init.	Adj.	Req.	Wait	Plan	Res.	**All**	Ext.	Time	Obj.	State	Exp.	Path	**All**	**Score**	W/T/L
Fixed harness	65.8	46.3	51.7	40.4	63.6	61.8	54.9	68.2	49.7	65.8	54.1	70.4	51.4	59.9	57.4	Ref.
Offline baselines: fixed at test time																
ExpeL	69.3	52.4	57.1	47.2	70.3	68.4	60.8	69.8	52.7	69.2	55.8	72.1	54.9	62.4	61.6	44.2/27.5/28.3
GEPA	74.1	51.8	58.7	48.3	71.6	69.9	62.4	75.2	55.4	72.1	57.3	75.2	53.1	64.7	63.6	50.7/24.4/24.9
EvoSkill	**74.2**	52.8	59.2	48.7	**74.1**	72.1	63.5	**77.2**	56.4	**75.1**	59.4	76.8	58.9	67.3	65.4	57.3/21.8/20.9
Meta-Harness	70.7	53.6	58.8	47.7	71.8	67.9	61.8	74.2	57.1	72.3	59.4	75.1	58.8	66.2	64.0	54.8/24.6/20.6
Online baselines: updated between events																
Dyn. Cheatsheet	69.2	51.3	54.5	45.4	68.2	67.3	59.3	69.8	50.7	68.9	55.1	71.9	55.1	61.9	60.6	39.4/28.7/31.9
A-MEM	67.7	52.6	58.4	47.9	65.7	65.8	59.7	69.1	56.4	69.2	58.8	70.8	54.2	63.1	61.4	43.1/25.4/31.5
AWM	70.4	51.2	56.6	46.1	71.3	**72.2**	61.3	72.7	52.8	71.9	57.7	75.5	57.6	64.7	63.0	48.7/27.4/23.9
ACE	68.7	52.6	56.5	47.8	67.7	64.6	59.7	70.3	54.2	69.8	57.1	70.8	56.4	63.1	61.4	40.2/24.7/35.1
ReasoningBank	72.3	57.4	62.4	54.1	72.2	70.5	64.8	72.8	59.1	72.6	62.2	74.7	56.9	66.4	65.6	58.4/22.7/18.9
TTHE	72.8	53.6	58.7	49.8	72.9	68.7	62.8	71.8	56.4	72.3	59.1	74.4	57.2	65.2	64.0	49.8/25.3/24.9
Harness-R1	73.3	57.1	61.7	51.2	72.3	70.2	64.3	73.8	57.6	72.4	60.9	75.8	53.7	65.7	65.0	55.3/24.8/19.9
MemEvolve	73.2	61.4	66.1	55.3	72.8	68.4	66.2	72.8	62.2	71.9	62.9	73.1	59.1	67.0	66.6	62.4/20.7/16.9
Study-specific controls																
SA-Reflection	69.7	53.8	59.6	50.7	69.5	66.6	61.7	71.2	57.4	69.8	60.2	72.9	**59.3**	65.1	63.4	50.4/24.7/24.9
Text-Gain	73.3	59.1	64.4	54.5	72.3	68.2	65.3	72.8	60.7	72.6	62.5	74.7	56.9	66.7	66.0	60.3/24.8/14.9
FLARE	**74.2**	**65.7**	**69.8**	**59.8**	**74.1**	**72.2**	**69.3**	**77.2**	**64.7**	**75.1**	**64.8**	**77.1**	**59.3**	**69.7**	**69.5**	70.3/19.6/10.1

Evaluation protocol. On Streaming^2-Bench, all methods share the executor, tools, event streams, scoring rules, and legally released information, including forecast lineage. Text-based methods receive descriptions of these relationships. We match budget limits separately for offline optimization, task execution, and event updates. Overall weights event scores equally after balancing the two task types within each event. W/T/L compares those scores against the fixed harness, rather than counting beneficial or harmful edits. Individual edits are evaluated by paired later-event gains under Section 3.3, so their effects are distinct from performance over a continuously evolving stream.

5.2 Comprehensive Evaluation

FLARE achieves the highest Overall score of 69.5, exceeding MemEvolve, the strongest existing baseline, by 2.9 points (Table 1). Its margin over Text-Gain is 3.5 points. It outperforms the fixed harness on 70.3% of events but scores lower on 10.1%.

5.3 Ablation Study

We separate representation design from its use by the Evolver and Predictor. Relation encoding improves Overall by 1.6 points over text conditioning, and conditional residuals add another 1.9 points (Table 2(a)). With the Encoder fixed, conditioning both generation and prediction outperforms conditioning either module alone (Table 2(b)). The panels use different fitting protocols, so their absolute scores are not directly comparable.

Reinforcement learning with measured training gains produces a positive-gain edit in 80% of candidate pools, versus 72% for continued SFT (Table 3(a)). Both variants produce valid substantive edits in 96% of attempts.

To isolate selection, each selector processes the five generators' pools separately. We weight generators equally. If a selector keeps the harness, that opportunity contributes zero gain. On these pools, mean decision gain is 1.90 points for the Edit-Gain Predictor and 1.40 for the text predictor

Table 2: Representation ablations with matched information and 0–100 scores. Panel a retrains affected components, using text when both are absent. Panel b freezes the Encoder: ✓ denotes Encoder conditioning and × denotes text. Compare within panels; bold marks their column maxima.

(a) Representation components

Graph enc.	Residual	Action	Spatial	Overall
×	×	65.3	66.7	66.0
✓	×	67.2	68.0	67.6
✓	✓	**69.3**	**69.7**	**69.5**

(b) Encoder conditioning

Evolver	Predictor	Action	Spatial	Overall
×	×	64.8	66.8	65.8
✓	×	66.6	67.4	67.0
×	✓	67.0	68.0	67.5
✓	✓	**68.4**	**68.6**	**68.5**

Table 3: Generation with eight attempts per state and selection on identical pools. Gains are mean paired later-event differences on 0–100 scores. Coverage is the weighted percentage of state-specific pools containing a positive-gain edit; Valid is the percentage of attempts yielding valid substantive edits. Best gain averages pool maxima including no-op. Harm counts negative-gain commits as a percentage of commits. Gain and Commit use all opportunities, with no-op gain zero. Audit-best uses private gains and is not deployable. Bold marks best deployable values in directed columns.

(a) Candidate generation

Generator	Coverage ↑	Best gain ↑	Valid ↑
Harness-R1	64.0	+2.20	92.0
SA-Reflection	58.0	+1.80	87.0
Text-Gain	70.0	+2.60	94.0
Lineage-Conditioned Evolver			
Continued SFT	72.0	+2.70	**96.0**
SFT + gain RL	**80.0**	**+3.40**	**96.0**

(b) Edit selection

Selector	Gain ↑	Harm ↓	Commit
Keep harness	0.00	–	0.0
Random	+0.30	35.0	84.0
LLM critic	+0.95	23.0	64.0
Text predictor	+1.40	15.0	58.0
Edit-Gain Predictor	**+1.90**	**9.0**	60.0
Audit-best	+2.54	0.0	68.8

(Table 3(b)). Negative-gain edits fall from 15% to 9% of commitments, while commitments rise from 58% to 60% of update opportunities.

5.4 Transfer Evaluation

We evaluate four leave-one-hazard-out folds, excluding all event groups containing the held-out hazard from offline learning, model selection, and threshold tuning (Appendix G). For groups containing several hazards, exclusion also removes their other-hazard records. Test-time harness updates remain allowed, so this evaluates transfer with online adaptation.

With hazards weighted equally, FLARE scores 64.40 versus 62.30 for the text-conditioned editor, a 2.10-point gain (Table 9). Its gains over that editor are 2.8, 3.0, and 2.6 points for cyclones, heavy rainfall, and floods. For wildfires, both methods score 54.2, exceeding the fixed harness at 51.0. FLARE thus improves over the fixed harness across all four hazards and matches or exceeds the text editor.

6 Conclusion

Streaming^2 provides a reusable benchmark connecting decisions under gradually arriving evidence with persistent harness updates across disaster events. Diagnostics show that lineage annotations reduce harmful updates in a baseline agent. Even with these annotations, gains on completed tasks do not always transfer to later events. FLARE uses forecast content and lineage to generate complete updates and select them by predicted future gains over making no change. On held-out events, FLARE improves overall task scores, although gains vary across subtasks and hazards and some events regress.

Ethics Statement

This retrospective evaluation does not establish readiness for autonomous emergency deployment. Operational use requires domain expert oversight. Data reuse must respect source licenses and protect sensitive locations.

Reproducibility Statement

Appendices A and B specify task scoring, information boundaries, and paired execution. Appendices C and D describe training targets, parameter updates, model architecture, and controlled comparisons.

AI Use Statement

AI tools were used for text polishing and auxiliary dataset annotation. The final review and validation of annotated data were completed manually by expert authors, who are responsible for the final content.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning, 2026. URL https://arxiv.org/abs/2507.19457.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.02766.

Pengyu Dai, Weihao Xuan, Junjue Wang, Hongruixuan Chen, Jian Song, Yafei Ou, and Naoto Yokoya. Experience-Driven Multi-Agent Systems Are Training-free Context-aware Earth Observers, 2026. URL https://arxiv.org/abs/2602.02559.

Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating Conflicting Data: The Role of Source Dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009. doi: 10.14778/1687627.1687690. URL https://doi.org/10.14778/1687627.1687690.

Jiarun Fu, Lizhong Ding, Sida Chen, Honglei Xin, Chunhui Zhang, Pengqi Li, Qiuning Wei, Ye Yuan, and Guoren Wang. CHILL-Harness: Counterfactual Harness Learning for Efficient Reasoning in Long-Horizon Agents, 2026. URL https://arxiv.org/abs/2607.25825.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft Adaptive Policy Optimization, 2025. URL https://arxiv.org/abs/2511.20347.

Tilmann Gneiting. Making and Evaluating Point Forecasts, 2010. URL https://arxiv.org/abs/0912.0902.

Shashwat Goel, Nikhil Chandak, Arvindh Arun, Ameya Prabhu, Steffen Staab, Moritz Hardt, Maksym Andriushchenko, and Jonas Geiping. FutureSim: Replaying World Events to Evaluate Adaptive Agents, 2026. URL https://arxiv.org/abs/2605.15188.

Maram Hasan, Aman Verma, Savitra Roy, Hariseetharam Gunduboina, Daksh Jain, Muhammad Haris Khan, Subhasis Chaudhuri, and Biplab Banerjee. GeoDisaster: Benchmarking Orchestrated Agents for Operational Disaster Geo-Intelligence, 2026. URL https://arxiv.org/abs/2606.17246.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering, 2024. URL https://arxiv.org/abs/2402.07630.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021. URL https://arxiv.org/abs/2106.09685.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous Graph Transformer, 2020. URL https://arxiv.org/abs/2003.01332.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Jun Nie, Yonggang Zhang, Jun Song, Qianshu Cai, Dahai Yu, Yike Guo, Xinmei Tian, and Bo Han. Tthe: Test-time harness evolution, 2026. URL https://arxiv.org/abs/2607.08124.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2026. URL https://arxiv.org/abs/2509.25140.

Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. WeatherBench 2: A benchmark for the next generation of data-driven global weather models, 2024. URL https://arxiv.org/abs/2308.15560.

Shuai Shao, Kangning Zhang, Qingyao Li, Shijian Wang, Hao Wang, Wenxiang Jiao, Yuan Lu, Yi Guo, Weiwen Liu, and Weinan Zhang. Harness-R1: Learning to Edit Executable Runtime Harnesses from Agent Failure Trajectories, 2026. URL https://arxiv.org/abs/2608.02276.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory, 2025. URL https://arxiv.org/abs/2504.07952.

Sumanth Varambally, Marshall Fisher, Jas Thakker, Yiwei Chen, Zhirui Xia, Yasaman Jafari, Ruijia Niu, Manas Jain, Veeramakali Vignesh Manivannan, Zachary Novack, Luyu Han, Srikar Eranky, Salva Rühling Cachay, Taylor Berg-Kirkpatrick, Duncan Watson-Parris, Yi-An Ma, and Rose Yu. Zephyrus: An Agentic Framework for Weather Science, 2026. URL https://arxiv.org/abs/2510.04017.

Junjue Wang, Weihao Xuan, Heli Qi, Pengyu Dai, Kunyi Liu, Hongruixuan Chen, Zhuo Zheng, Junshi Xia, Stefano Ermon, and Naoto Yokoya. Can LLM Agents Respond to Disasters? Benchmarking Heterogeneous Geospatial Reasoning in Emergency Operations, 2026. URL https://arxiv.org/abs/2605.11633.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024. URL https://arxiv.org/abs/2409.07429.

Chuyang Wei, Maohang Gao, Zhixin Han, Kefei Chen, Yu Zhuang, Minghang Zhu, Haoxiang Guan, Yanzhi Zhang, Yilin Cheng, Xiren Zhou, Huanhuan Chen, Jian Li, Jiyan He, Yu Shi, Yitong Duan, and Shuxin Zheng. Harnessing Pre-Resolution Signals for Future Prediction Agents, 2026. URL https://arxiv.org/abs/2604.15719v3.

Ke Xu, Han Xu, Xinran Chen, Yuqian Wang, Zhixuan Li, Xiaojian Liu, Changwo Wu, Jianqiang Xia, and Yuchen Li. STAMP: Provenance-Guided Credit Assignment for Deep Search Agents, 2026. URL https://arxiv.org/abs/2607.11172.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents, 2025. URL https://arxiv.org/abs/2502.12110.

Min Yang, Jinghua Piao, Xu Xia, Xiaochong Lan, Jiaju Chen, Yongshun Gong, and Yong Li. SkillMaster: Toward Autonomous Skill Mastery in LLM Agents, 2026. URL https://arxiv.org/abs/2605.08693.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems, 2025. URL https://arxiv.org/abs/2512.18746.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models, 2026. URL https://arxiv.org/abs/2510.04618.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL https://arxiv.org/abs/2308.10144.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. LifelongAgentBench: Evaluating LLM Agents as Lifelong Learners, 2025. URL https://arxiv.org/abs/2505.11942.

A BENCHMARK CONTRACTS AND INFORMATION BOUNDARIES

Event packages. Each event package identifies the event group, region, time window, data versions, decision instances, tools, and feedback rules. Its evidence inventory includes content hashes, variables, units, coverage, licenses, and justified availability. The public task view is generated by the host from the current role, decision instance, current event time, and request receipts. The agent does not choose these authorization conditions. Private Oracle data and future directories are excluded from the public view. A source relation becomes visible only when both endpoints and the relation itself are available.

Streaming execution. Core evaluation events contain both task types, which can be interleaved as evidence arrives. Each task instance is a decision instance (DI), with a decision time, deadline, tool-call budget, and required output. At each decision, agents can access only evidence already available under the release protocol, including responses to permitted requests. A released forecast may describe future conditions, while later observations remain inaccessible until their release. Agents can keep temporary notes of their spatial results for later decisions within the same event. Forecast-verification feedback follows its own release schedule.

The harness contains execution code, applicability conditions, and guidance. The executing model and tools remain fixed, while temporary notes reset between events. Events follow a fixed experimental order even when their historical time windows overlap.

Time and historical reconstruction. An asset can have different issue, valid, observation, publication, and availability times. Download time records acquisition by the researchers, not historical public availability. A later correction to a historical observation is a distinct version. For directly documented availability, the corresponding archive evidence is retained. For reconstructed availability, the rule, source justification, uncertainty, and fixed parameters are retained. Unknown availability does not default to immediate access. Requests for future products can remain pending; a request does not create a new satellite observation or field report.

Task and event scores. Task scores evaluate complete action plans and structured spatial results using separate criteria. For an action DI d, let $\mathcal{P}_{e,d}$ contain its predefined acceptable plan patterns. The indicator $m_{e,d,p} \in \{0,1\}$ records whether the submitted plan satisfies all conditions of pattern p. For a spatial DI, let $F_{e,d}$ contain its predefined scored fields, with field scores $s_{e,d,f} \in [0,1]$. Both sets are nonempty for their respective task types. The task score before process gates is

$$S_{e,d} = \begin{cases} \max_{p \in \mathcal{P}_{e,d}} m_{e,d,p}, & d \in D_e^{\text{action}}, \\ \dfrac{1}{|F_{e,d}|} \sum_{f \in F_{e,d}} s_{e,d,f}, & d \in D_e^{\text{spatial}}. \end{cases} \tag{3}$$

For either task type, we apply $u_{e,d} = G_{e,d}^{\text{exec}} G_{e,d}^{\text{evidence}} G_{e,d}^{\text{policy}} S_{e,d}$. These binary gates check executable output, legal evidence access, and public hard policies. We average DIs within each task type and then give the task types equal weight:

$$S_e(H) = \tfrac{1}{2}\left(S_e^{\text{action}}(H) + S_e^{\text{spatial}}(H)\right), \qquad S_e^k(H) = \frac{1}{|D_e^k|} \sum_{d \in D_e^k} u_{e,d}(H). \tag{4}$$

This convention prevents a larger number of spatial questions from overwhelming action evaluation. It is a fixed measurement choice, not a learned reward designed to favor FLARE. Other agents can use the same input, tool, output, and scoring interfaces. The equations use task scores in $[0,1]$. Reported scores and gain differences are multiplied by 100.

Spatial scoring. Scalar, categorical, point, and polygon outputs use frozen correctness checks with task-specific tolerances. Set outputs use one-to-one object matching and F1. For a nonzero denominator, $F_1 = 2\text{TP}/(2\text{TP} + \text{FP} + \text{FN})$. Both-empty sets and unavailable reference coverage have explicit task rules. Route checks include the required endpoints and geometry constraints; safety tasks additionally require the specified traversability conditions. Coordinate systems, nodata handling, geometric validity, and allowed repairs are fixed before evaluation. Dictionary outputs use a declared aggregation level to avoid counting both a parent field and all its children.

Action patterns and support. Each acceptable pattern checks a complete plan against the information available at the decision time, without using later disaster outcomes. Its conditions jointly cover action type, target, parameters, timing, required requests, caveats, and submission or execution, together with the necessary evidence support. Satisfying one complete pattern earns one point before process gates, regardless of which acceptable plan the agent chooses. Satisfying only some conditions, or combining satisfied conditions from different patterns, earns no partial credit. Waiting, abstaining, or escalating is not automatically correct. Finding a cited asset is distinct from verifying that its type, time window, coverage, and quality support the asserted field. An unsupported statement and actual unauthorized data access are separately audited.

Feedback and evolution timing. The event package distinguishes required feedback from optional later feedback. Only legally released items enter the evolution snapshot. All DIs must finish before the single evolution opportunity opens. A `no-op` closes that opportunity just as a committed transaction does. Additional late feedback does not silently create another opportunity for the same event. The scorer may evaluate more information than the agent is authorized to receive at that time.

B Harness Transactions and Paired Replay

Table 4: The four fixed entry points and their permitted roles. Hook code can change through a complete event-level transaction; entry points and host permissions remain fixed.

Hook	Role
`on_init`	Supply reusable guidance and initialize the event notebook.
`make_pre_hint`	Add guidance before the next decision using permitted state.
`on_before_action`	Check or adjust a proposed action within host permissions.
`on_post_step`	Process step feedback and update temporary event notes.

An edit binds to the complete current harness version. The transaction has explicit retain, replace, and delete semantics, even when unchanged code is materialized rather than regenerated. Deleting the last hook is a substantive change; it is not `no-op`. A candidate is validated and committed atomically. Partially installing it would produce a different object from the one the Gain Predictor scored. Failures during execution follow a frozen fallback rule and remain in the trace. Restoring earlier code is a new transaction at the next lawful opportunity, not an unrecorded rollback that erases an observed regression. A frozen tie-breaking rule resolves equal predicted gains among valid candidates.

Edit selection. Let $\mathcal{D}_t$ be the finite set of valid candidates excluding `no-op`. For nonempty $\mathcal{D}_t$, we select δ_t^* using the fixed tie-breaking rule:

$$\delta_t^* = \arg\max_{\delta \in \mathcal{D}_t} \widehat{\mu}_t(\delta),$$
$$H_{t+1} = \begin{cases} H_t^{\delta_t^*}, & \mathcal{D}_t \neq \varnothing \text{ and } \widehat{\mu}_t(\delta_t^*) > \tau, \\ H_t, & \text{otherwise.} \end{cases} \tag{5}$$

Paired execution and gain labels. An edit δ is a version-bound transaction that materializes $H_t^\delta = H_t \oplus \delta$. The evaluator compares H_t^δ and H_t on the next K eligible events within the same data split. For target event e_j and repeat r, it records

$$\Delta_{t,j,r}(\delta) = S_{e_j,r}(H_t^\delta) - S_{e_j,r}(H_t), \qquad \bar{\Delta}_t(\delta) = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{R_j}\sum_{r=1}^{R_j}\Delta_{t,j,r}(\delta). \tag{6}$$

$$\bar{S}_e(H) = \frac{1}{R_e}\sum_{r=1}^{R_e} S_{e,r}(H).$$

Each target execution starts from its arm's fixed harness and fresh temporary state. Neither arm makes further persistent edits inside the label window. The arms share the event world, release

rules, tools, budgets, scorer, and specified randomization protocol. Their requests, actions, and trajectories may differ; these differences are how the edit takes effect.

Edit-credit records retain the legal source snapshot, transaction, paired runs, and positive, zero, or negative differences. Task failures remain valid outcomes; infrastructure failures are missing measurements. Incomplete windows never extend into another split or expose future outcomes to the source decision.

For source t, target event j, and repeat r, paired records retain the baseline and modified run identifiers, event scores, template-specific scores, validity, and component versions. The paired conditions cover the external event world, release rules, tools, executor, budgets, and randomization. They do not force identical actions. Sharing a baseline across candidates requires a complete cache key, including initial temporary-state rules. A cached score with a changed model, task, Oracle, or randomization is not a valid shared baseline.

The credit target is the distribution of paired score differences under the declared protocol. Changing the coupling can change difference quantiles even when arm-wise expectations remain unchanged. A common seed is therefore not a universal variance-reduction guarantee. The no-op decision has zero structural gain; this identity does not assert that two independent stochastic reruns return equal scores.

Table 5: Lineage diagnostics on Streaming^2-Evo validation events. Panel a averages gains and commit rates over all opportunities, with no-op gain zero. Panel b uses edits retained with annotations. Gains are mean paired differences on 0–100 scores. Harmful means negative mean later gain. Rates are percentages. Bold marks better directed values.

(a) Lineage visibility and retained-edit quality

Lineage	Commit rate	Harmful / opportunity ↓	Harmful / commit ↓	Later gain ↑
Provided	59.2	**31.3**	**52.8**	**+0.82**
Hidden	68.8	42.9	62.4	+0.23

(b) Original-task repair and later gains of commits with lineage

Original gain	Later gain	Original gain > 0 (%)	Later gain ≤ 0 (%) given original gain > 0 ↓
+3.59	+1.38	80.0	65.0

Diagnostic aggregation. Table 5 applies the same source-event-balanced weights to both panels. These diagnostics use the baseline's own candidate pools, whereas Table 3 evaluates each selector on every generator's pools separately. Panel (a) computes Commit rate and mean Later gain over all update opportunities, assigning zero gain to `no-op`. The two harmful-update rates count commits with strictly negative mean later gain and use the denominators shown in the table. Panel (b) uses the Provided arm's complete commits, with the same weights conditioned on commitment. Its last column further conditions on positive original-task gain and measures the fraction with zero or negative mean later gain.

Reporting edit regressions. For a measured candidate window, the event-balanced negative-difference frequency is

$$\mathrm{Reg}_t(\delta) = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{R_j}\sum_{r=1}^{R_j}\mathbf{1}[\Delta_{t,j,r}(\delta) < 0]. \tag{7}$$

This frequency is distinct from the magnitude of negative gains. An aggregate audit must state which committed edits have complete eligible windows and report audit coverage. A fixed harness or a stream with no commits has no edit-conditioned regression estimate. The main closed-loop task score remains defined for such a system.

C TRAINING BOUNDARIES AND STATISTICAL TARGETS

Training sequence. We initialize FLARE to evaluate and generate harness edits, first fixing the reference for content differences and then learning from measured gains. To establish that reference, we train the conditional predictor P_η to predict product features from related products and then freeze it. For initial gain supervision, we collect harness states using a fixed policy that proposes edits from available experience and carries retained changes across training events. We label candidate edits to these harnesses through paired runs on later training events under the protocol in Section 3.3. To learn which edits improve later performance, we fit the Gain Predictor to these gains while jointly updating the Encoder's learnable components. We next train the Evolver and its input projection to generate complete edits, keeping the Encoder and Gain Predictor fixed. For each state, the target is the evaluated valid candidate with the highest positive average gain, or `no-op` if none improves on the unchanged harness. Later outcomes determine the training targets but never enter the inputs available when the candidates were proposed.

To learn from its own updates after initialization, FLARE collects new harness states and candidate edits as it runs through training events. For each valid sampled edit, isolated paired runs on later training events provide a measured gain as its reward, rather than the Gain Predictor's estimate. Using these rewards, Soft Adaptive Policy Optimization (SAPO; Gao et al., 2025) updates the Evolver and its input projection, with the Encoder and Gain Predictor fixed. Between policy-update blocks, we update the Gain Predictor and the Encoder's learnable components using accumulated gain records to reflect the newly encountered states. To adapt generation to the updated event representations z_t, we briefly repeat supervised fine-tuning of the Evolver and its input projection. New collection then starts after all events already used for learning or model selection, including those that supplied gain labels.

To choose when FLARE should retain edits, we freeze all learned components and select the commit threshold τ using validation task performance. Because different values of τ can produce different harnesses and later experience, we compare them through complete validation runs from the same initial harness. During testing on Streaming^2-Bench, all learned parameters and τ remain fixed, while the harness can evolve using feedback released under the streaming protocol. Private outcomes from paired runs on later test events are used only for evaluation and never guide candidate generation or retention.

Table 6: Parameter updates within each optimization block. Credit-fit blocks recur between policy-update blocks. Fixed feature encoders, the task executor, tools, scoring, and host permissions remain frozen throughout.

Component	Content pretrain	Credit fit	SFT / RL	Test
Conditional predictor	Update	Freeze	Freeze	Freeze
Residual gate, HGT, readout	–	Update	Freeze	Freeze
Evolver adapters and projection	–	Freeze	Update	Freeze
Gain-predictor fusion and output heads	–	Update	Freeze	Freeze

Shared-verification links. We construct shared-verification links from the observation records used in released forecast-verification feedback. Two distinct forecast nodes are connected when their feedback records refer to the same version of at least one observation record. Matching uses identifiers of the actual observation records and their versions, with spatial and temporal scope preserved, rather than dataset or file identifiers alone. Each link is represented in both directions and is added only after both forecast nodes are available and their supporting feedback is released. Each forecast's verification results enter its own node feedback features f_v, while shared observation identities determine the graph links. Reference eligibility for $\mathcal{R}_t(v)$ depends only on the upstream or earlier-version criteria, independently of shared-verification links. The predictor uses fixed reference features, excluding shared-verification links, verification feedback, and HGT outputs. Unknown verification relationships remain unmarked and are not treated as evidence of independence.

Content and residual fusion. We project the conditional residual ϵ_v from Equation 2 without a bias, apply a learned gate, and retain the absolute content:

$$g_v = \sigma(\mathrm{MLP}[c_v; \widehat{c}_v; m_v; \ell_v; f_v]),$$
$$h_v = \mathrm{MLP}[c_v; g_v \odot W_{\mathrm{type}(v)}\epsilon_v; m_v; \ell_v; f_v; b_v]. \tag{8}$$

The features ℓ_v describe actual evidence use; f_v contains released feedback; b_v records availability and missingness. When valid references are absent, the residual contribution is zero without evaluating its predictor or gate, with an explicit unavailable flag. The conditional predictor minimizes event-balanced masked squared error in a fixed feature space. Nodes without valid reference sets do not supply this prediction target. Missing dimensions are omitted, not treated as true zeros. The predictor's input excludes target content and later feedback, even when these are present elsewhere in the stored event package. Legal released feedback can subsequently enter the residual gate and event encoder. This separation avoids learning a trivial content reconstruction from the answer itself.

Edit encoding and gain supervision. The edit representation in Section 4.2 is

$$c_\delta = \mathrm{Embed}_{\mathrm{frozen}}\big(\mathrm{Serialize}(H_t, \delta, H_t^\delta)\big). \tag{9}$$

Alongside the mean used for edit selection, three auxiliary heads predict gain quantiles $\widehat{q}_\alpha$ for diagnostics. Let $\mathcal{A} = \{.25, .50, .75\}$ and $\rho_\alpha(a) = a(\alpha - \mathbf{1}[a < 0])$. The credit loss is

$$\mathcal{L}_{\mathrm{credit}} = (\widehat{\mu} - \bar{\Delta}_t)^2 + \frac{\lambda_q}{|\mathcal{A}|}\sum_{\alpha\in\mathcal{A}}\frac{1}{K}\sum_{j=1}^{K}\frac{1}{R_j}\sum_{r=1}^{R_j}\rho_\alpha(\Delta_{t,j,r} - \widehat{q}_\alpha). \tag{10}$$

Raw paired differences supervise quantiles; replacing them with repeat-averaged values would change the target distribution. This loss updates the trainable encoder and value network, without differentiating through discrete edits or the frozen text encoder. It scores the complete transaction, allowing interactions between hooks rather than assuming additive hook values.

For gain prediction, squared loss targets the conditional mean. For a protocol-defined gain Z and input X, the identity

$$\mathbb{E}[(Z - f(X))^2 \mid X] = \mathrm{Var}(Z \mid X) + \big(f(X) - \mathbb{E}[Z \mid X]\big)^2 \tag{11}$$

explains that target without guaranteeing finite-sample accuracy. Pinball loss targets a quantile of the same event-weighted gain distribution (Gneiting, 2010). At atoms, a valid α quantile satisfies $\Pr(Z < q_\alpha) \le \alpha \le \Pr(Z \le q_\alpha)$. Thus a large mass at zero does not imply that exactly an α fraction must lie strictly below the predicted quantile. Any deterministic correction for quantile crossing is fixed before evaluation and assessed after correction. Neither ordered quantiles nor aggregate coverage establishes conditional calibration. Quantiles are diagnostic outputs and do not impose an additional acceptance gate.

Collected states and labels. A fixed initial collector generates real cumulative harness states. Candidate labels are gathered in isolated branches, which do not overwrite the collection stream with whichever private branch later scores best. After training updates, new collection begins beyond the maximum event index consumed by normalization, content fitting, credit fitting, SFT selection, policy rewards, and model selection. This prefix rule concerns information used by the learned system, not whether the researchers have downloaded the archive. Old labels remain usable if all execution conditions stay identical; changing only the encoder permits recomputing states from original snapshots. Old policy samples do not become fresh on-policy samples after this refresh.

Policy reward and candidate groups. All valid candidates in the original sampled group receive measured gains, regardless of the Gain Predictor's ordering. Teacher proposals belong to a separate supervised pool. If `no-op` participates in a policy-gradient group, it must be sampled by that policy rather than inserted as an extra generated response. The invalid-transaction penalty is an optimization device and is never reported as benchmark task score. A shared subtraction of the same baseline does not alter group-centered advantages among otherwise comparable candidates. Paired gains remain essential for absolute edit credit and evaluation; subtracting a baseline is not itself a new optimization algorithm.

D IMPLEMENTATION DETAILS

Table 7 summarizes the model and architecture configuration. We use Qwen3.8-27B for two isolated roles: the frozen executor and the adapted evolver. The task executor runs in BF16 on A100 80GB GPUs. The Evolver's continuous prompt vectors precede the embeddings of the current harness and available experience text in a single input sequence. The modification encoder uses frozen Qwen3-Embedding-4B. The complete serialization must fit the input budget without truncation. We train on Streaming^2-Evo's training split and select checkpoints on its validation split.

Table 7: Model and architecture configuration.

Item	Configuration
Relation predictor	Two attention blocks, four heads
Residual projection / event state	256 / 512 dimensions
Graph encoder	Two HGT layers with type-wise pooling
Evolver conditioning	Eight soft prompt positions
Evolver adaptation	LoRA rank 64
Joint modification embedding	2560 dimensions
Gain-predictor fusion	$3072 \rightarrow 1024 \rightarrow 256$; mean and three quantile heads
Policy optimization	SAPO with measured paired-gain rewards
Generation attempts per opportunity M	Eight

Training schedule. We separate content pretraining from four forward collection phases so that newly fitted components only encounter events beyond their training targets. The first 80 training events establish the content reference: events 1–64 fit feature normalizers and P_η, and events 65–80 select its checkpoint by content-prediction error. We then freeze both, and exclude these 80 events from edit collection. The remaining phases cover events 81–220, 221–360, 361–500, and 501–640. Each phase runs four collection trajectories through the same event sequence. The first 135 events permit candidate generation, while the final five supply later-event targets without generating further edits. Only windows with five eligible targets inside the phase receive complete gain labels. Paired branches never replace the actual collection trajectories.

The first phase uses a fixed text-based generator and critic, which select edits from legally available information. After its labels are complete, we fit the Encoder and Gain Predictor and initialize the Evolver by SFT. The three later phases use FLARE with a zero predicted-gain threshold and continue each trajectory's retained harness. Learned parameters remain fixed during each collection phase. At the end of each of these three phases, SAPO first updates the Evolver and its projection using that phase's original candidate groups. We then refresh the Encoder and Gain Predictor on accumulated labels, keeping feature normalizers frozen, and repeat SFT to adapt generation to the refreshed representation. The next collection phase starts beyond all events used by these labels. We save a checkpoint after the first phase's initialization and after each later phase's SAPO, credit refresh, and SFT refresh. Validation begins only after all four phases have ended.

Supervision records. Every valid substantive candidate in an original eight-attempt group receives a paired audit, regardless of its predicted score. A compact index retains up to 8,000 distinct source–candidate records for credit fitting. Records enter this index before their audit outcomes are read, using a fixed hash priority over source snapshots, current and candidate harnesses, and the execution contract. At each training boundary, the index retains the 8,000 smallest hash values among records collected so far, or all records if fewer are available. Evicted records remain in their original policy groups and SFT pools. Positive, zero, and negative gains remain eligible, and an incomplete audit leaves an unavailable slot rather than triggering outcome-based replacement. Exact duplicate records share storage, but their original sampled occurrences remain in the policy groups. SFT targets use complete original candidate pools, including audited candidates outside this compact index. A pool with a missing candidate audit supplies neither an SFT target nor a policy update; individually complete records remain usable for credit fitting. `no-op` supplies a policy outcome when sampled but is not a substantive edit in the 8,000-record inventory.

Optimization settings. Table 8 specifies separate learning rates for representation learning, supervised generation, and policy optimization. All stages use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01, and gradient-norm clipping at 1. Bias and normalization parameters receive no weight decay. Each fitting block starts a fresh optimizer and a cosine schedule with 5% warmup and a final learning rate of 10% of its peak. Content and credit epochs visit every eligible record once in a shuffled order, with losses weighted equally across source events and then within each event. SFT applies the same event balance to source states and averages completion-token cross-entropy within each response. The brief SFT refresh draws source events uniformly and then states uniformly within each event. Gain targets remain on the fixed $[-1, 1]$ task-score difference scale, with $\lambda_q = 0.01$ in Equation 10. This scale requires no fitted score normalizer. The three quantile heads remain diagnostic outputs and do not affect edit acceptance.

Table 8: Training settings. Batch sizes are global; an epoch is one pass through the indicated records. Learning rates refer to trainable components only.

Stage	Peak learning rate	Global batch	Duration
Content prediction	3×10^{-4}	256 product nodes	10 epochs
Initial credit fit	Encoder 10^{-4}; Predictor 3×10^{-4}	64 edit records	10 epochs
Credit refresh	Same as initial fit	64 edit records	3 epochs
Initial SFT	LoRA 10^{-5}; projection 5×10^{-5}	16 source states	2 epochs
SFT refresh	LoRA 2×10^{-6}; projection 10^{-5}	16 source states	64 updates
SAPO	LoRA 5×10^{-7}; projection 5×10^{-6}	4 groups of 8	1 pass

Policy objective. SAPO uses measured normalized gains for valid edits, zero for sampled `no-op`, and -1 for invalid transactions. The invalid penalty belongs only to training and never substitutes for a task score. Advantages center each original group of eight rewards and divide by its population standard deviation plus 10^{-6}. Constant-reward groups have zero advantages and are not resampled. We use token-level importance ratios, response-averaged token losses, and SAPO temperatures 1.0 and 1.05 for positive and negative advantages, with no KL penalty (Gao et al., 2025). A policy pass keeps every sampled occurrence, including duplicate and invalid outputs, and uses stored behavior log probabilities and conditioning versions. Mini-batches contain groups from distinct source events, with equal total loss weight per source event. Old groups are not reused after representation refresh. A missing paired reward excludes its entire policy group without zero imputation.

Adaptation and generation. The Evolver uses LoRA rank 64, scaling factor $\alpha = 128$, and zero adapter dropout. The input projection maps z_t to eight prompt vectors, and gradients through these vectors update that projection. Adapters cover language-model attention and MLP projections, including the hybrid backbone's linear-attention content projections. They exclude embeddings, the output vocabulary head, vision modules, and linear-attention decay gates. Evolver sampling uses temperature 1, top-$p = 1$, and no top-k restriction during collection and evaluation. Each response is a directly serialized edit transaction, with at most 4,096 generated tokens. Prompts permit 24,576 tokens including the eight soft positions, and a complete harness permits 4,096 language-model tokens. The joint modification serialization permits 32,768 tokens under the embedding model's tokenizer. We do not silently truncate inputs or edits. An overlong generated transaction is invalid; an unavailable source input keeps the harness unchanged and remains in complete-stream scoring. The frozen executor uses temperature 0.2 and top-$p = 1$, and the bootstrap critic uses greedy decoding.

Hardware and execution failures. Training uses four A100 80GB GPUs, BF16 computation, FP32 optimizer states, full parameter sharding, and activation checkpointing. Language-model micro-batches contain one sequence per GPU; gradient accumulation is four for SFT and eight for SAPO. Frozen executor and embedding inference are scheduled separately from policy optimization. Infrastructure failures permit two retries with identical inputs and seeds. Exhausted retries remain missing observations, whereas valid low-scoring executions retain their measured scores.

Evaluation settings. Each update opportunity permits eight raw generation attempts, including invalid, duplicate, and no-op outputs. Each substantive edit is evaluated on five later eligible events, with three paired repetitions per event. Sources without a complete window still contribute to complete-stream task scores; unavailable gains are not replaced with zero. After training, validation compares the four phase-end checkpoints and $\tau \in \{0, 0.25, 0.5, 1, 1.5, 2\}$ in 0–100 score points. Each combination runs three complete validation streams from the common initial harness. We select the highest mean task score, breaking exact ties by the larger threshold and then the earlier checkpoint. The implementation divides thresholds by 100, giving a normalized range from 0 to 0.02. Testing uses the selected frozen checkpoint and threshold in five complete streams with independent harness resets. These repetitions vary runtime randomness, not training initialization.

We use master seed 17, collection seeds $\{101, 202, 303, 404\}$, validation seeds $\{503, 509, 521\}$, and test seeds $\{601, 607, 613, 617, 619\}$. Paired repetitions use seeds $\{701, 709, 719\}$, shared between unchanged and edited arms. Per-call seeds combine these bases with the role, phase, trajectory, event, source snapshot, and call position. Paired executor seed keys omit candidate and arm identity to preserve common external randomness. Execution records retain separate identities for both arms.

For each leave-one-hazard-out fold, target-related event groups are excluded before normalization, content fitting, collection, and model selection. All learned components are fitted again using the remaining source-domain data. For N remaining training events, phase boundaries become $\lfloor N/8 \rfloor$, $\lfloor 11N/32 \rfloor$, $\lfloor 18N/32 \rfloor$, $\lfloor 25N/32 \rfloor$, and N. The content prefix uses an 80/20 fitting/selection split, and each collection phase reserves its final five events for labels. A fold without enough eligible events is reported as unavailable rather than weakening exclusion.

Computational cost. The relevant cost includes collecting edit labels as well as using the learned selector. With M_{valid} edits, K target events and constant repeats $R_j = R$, baseline reuse requires $(M_{\text{valid}} + 1)KR$ target-event executions. This count excludes source collection, retries, and neural training. Reuse is valid only when the source harness, event assets, executor, tools, budgets, and randomization match. At test time, the system pays for candidate generation and value prediction but does not replay private future events to choose an edit. The four phases permit at most 2,160 source states and 17,280 raw attempts across the four trajectories. Auditing every attempt as a valid edit would require at most $291{,}600$ target-event executions with baseline reuse, before retries. The 8,000-record supervision index is therefore not the total collection budget.

Task column abbreviations. In Table 1, Init., Adj., Req., Wait, Plan, and Res. denote initial response selection, response adjustment, evidence request selection, waiting/substitution, response plan selection, and resource allocation. Ext., Time, Obj., State, Exp., and Path denote disaster extent, temporal change, affected objects, object state, exposure, and connectivity/path analysis.

Baseline adaptation boundaries. The fixed harness makes no cross-event updates. Our adaptations of ExpeL (Zhao et al., 2024), GEPA (Agrawal et al., 2026), EvoSkill (Alzubi et al., 2026), and Meta-Harness (Lee et al., 2026) learn persistent artifacts offline and keep them fixed during testing. Dynamic Cheatsheet (Suzgun et al., 2025), A-MEM (Xu et al., 2025), AWM (Wang et al., 2024), ACE (Zhang et al., 2026), ReasoningBank (Ouyang et al., 2026), TTHE (Nie et al., 2026), and Harness-R1 (Shao et al., 2026) update persistent state between events. Our MemEvolve (Zhang et al., 2025) adaptation learns its memory architecture offline and updates its contents online. These settings describe our adaptations, rather than the full scope of the original methods. Both Source-aware Reflection and Text-conditioned Future-Gain Editor receive the same legally visible forecast relationships as FLARE. Text-conditioned Future-Gain Editor also receives the same edit-gain supervision while replacing the graph representation with a matched text interface. A Harness-R1-style adaptation preserves learned executable editing while explicitly declaring its source-event reward and persistence adaptation. Budget accounting includes reflection, retrieval, candidate generation, and replay. Any additional probes or training examples count toward the corresponding budget. During testing, learned parameters and selected settings remain fixed, while persistent updates follow the release protocol in Section 3.3.

Controlled ablation protocols. For Table 2, all variants receive the same legal information, including forecast relationships, with matched supervision and budgets. Each row runs complete event

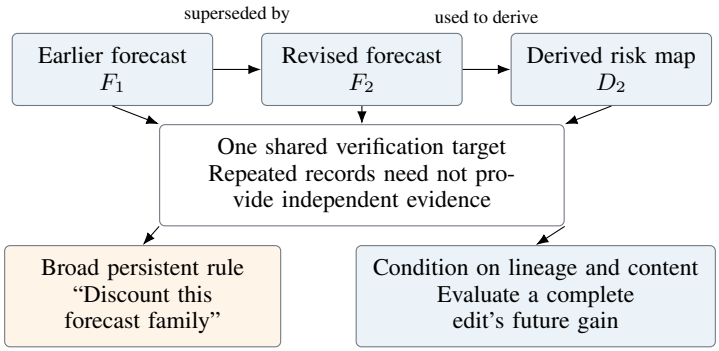

Figure 5: Why repeated forecast feedback need not justify a broad harness rule. Across the top, forecast F_2 replaces F_1 and produces derived map D_2. Arrows to the middle box indicate a shared verification target, so repeated records need not provide independent support. The lower branches contrast discounting the entire forecast family with using lineage and content to assess a complete edit's future benefit.

streams. The no-residual variant retains the same graph relations, original content, and gain supervision. Residual denotes the conditional-residual branch. In panel (a), both generation and prediction use the listed representation; affected components are retrained. Panel (b) fixes one Encoder learned only from training data across all variants, with matched downstream adaptation.

For Table 3, the continued-SFT and RL variants share an initial SFT checkpoint and an Encoder learned only from training data. The Encoder remains fixed, with matched training and replay budgets. RL rewards use measured paired gains. Valid is the percentage of all raw attempts that yield substantive edits passing the fixed checks. Coverage is the weighted percentage of shared snapshots with a positive-gain candidate. Best gain averages each pool's maximum gain or zero using the same snapshot weights. Each selector processes every generator's pools separately, without merging candidates across generators. Generator totals have equal weight, with common snapshot weights within each generator. Decision gain and Commit use all opportunities, with no-op gain zero. Harm divides total weighted negative-gain commits by total weighted actual commits. The learned predictors use matched training gain records and budgets. The LLM critic receives all legal information, and Random selects uniformly among distinct valid edits and one no-op. Audit-best selects the highest measured positive-gain candidate or keeps the harness unchanged. It uses private audit outcomes and is not deployable. In Table 3, bold marks the best deployable value in each column with an arrow. Commit has no preferred direction and is not highlighted.

E Forecast Feedback and Harness Evolution

Figure 5 illustrates related forecast feedback and the choice of a persistent update.

F Harness Evolution Across Events

Figure 6 traces how FLARE and the text-conditioned editor translate the same lineage-annotated experience into different rules for later events. Both methods start from the same harness under matched budgets. During Hurricane Ian, a wind-probability map and an arrival-time map display the same probability layer from one forecast run. The agent requests the same observation twice after consulting these maps, and released task feedback identifies the resulting budget waste. The text-conditioned editor responds by allowing only one request per location and forecast product family within an event. FLARE retains a narrower rule: merge requests for the same observation and location within a forecast run, but reassess when relevant content or timing changes.

During Hurricane Idalia, the agent has already requested evidence for a location when a revised forecast from the same product family arrives. The revision raises that location's wind risk, making an updated observation a priority before the next decision. The text-based rule blocks a further request, whereas FLARE retrieves the newly released observation and uses it to revise its response.

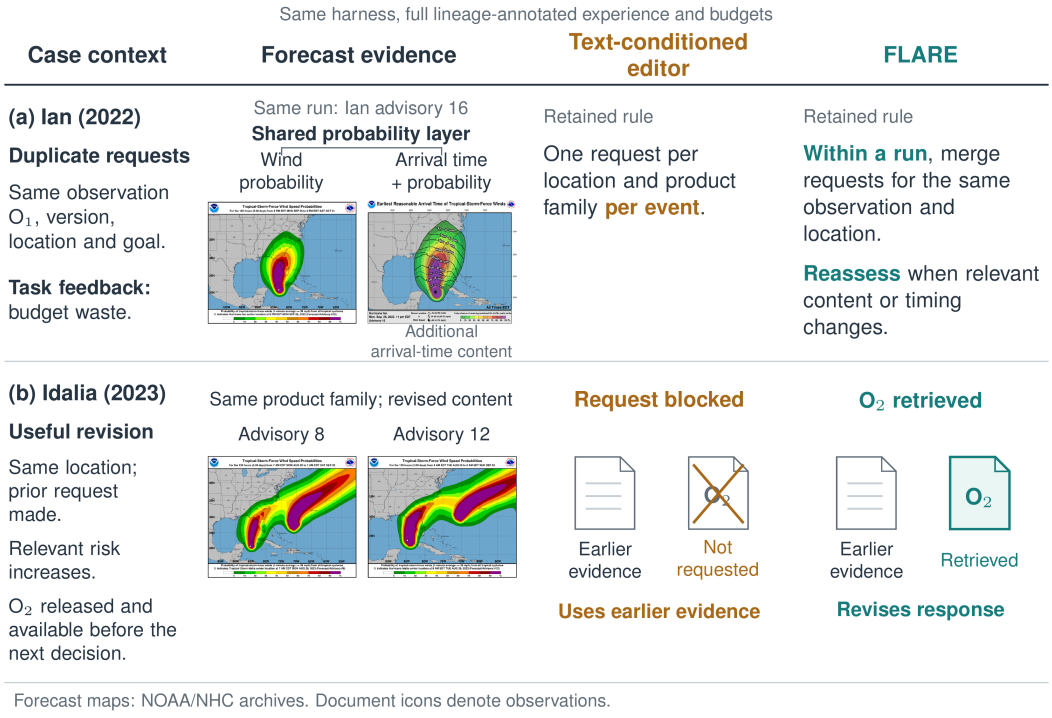

Figure 6: Harness rules learned from the same experience can respond differently to later revisions. In Figure 6(a), two Ian maps share a forecast probability layer. After duplicate observation requests, the text-conditioned editor limits requests per location and product family within an event. FLARE instead merges duplicates within a forecast run. In Figure 6(b), an Idalia revision raises local risk. The broad rule blocks another request, while FLARE retrieves the newly released observation O_2 and revises its response. Forecast maps: NHC archive images.

In this case, conditioning the retained rule on both lineage and content preserves a response to useful revisions.

G Transfer Protocol and Results

Table 9: Leave-one-hazard-out transfer. Target hazards are excluded from training, model selection, and threshold tuning. Overall scores use 0–100. Macro weights hazards equally. Bold marks maxima.

Method	Tropical cyclones	Heavy rainfall	Floods	Wildfires	**Macro**
Fixed harness	58.0	60.0	61.0	51.0	57.50
Harness-R1	62.5	63.5	65.0	51.8	60.70
SA-Reflection	61.5	62.8	64.0	52.5	60.20
Text-Gain	63.8	65.0	66.2	**54.2**	62.30
FLARE	**66.6**	**68.0**	**68.8**	**54.2**	**64.40**

Protocol and summary. To test transfer to hazards excluded from offline learning, we run four leave-one-hazard-out folds within the original Evo and Bench partitions. Each fold removes every group containing the target hazard from training, model selection, and threshold tuning, including that group's records of other hazards. Target streams start from the same harness; learned components stay frozen while legally released feedback can update the harness between events. Scores cover each full target sequence.

H RESOURCE INVENTORY

Table 10: Resource composition and training edit-credit records.

Unit	Count
Events	1,000
Action Selection instances	4,000
Spatial Analysis instances	6,000
All decision instances	10,000
Forecast records	160,000
Forecast records with verified lineage links	96,000
Lineage annotation coverage (%)	60
Satellite scenes	20,000
Station observations	5,000,000
Mean available forecast records per task	24
Mean available satellite scenes per task	6
Training edit-credit records	8,000