# TRAJFOUNDRY: EXPERIENCE-DRIVEN TRAJECTORY SYNTHESIS FOR MULTIMODAL AGENTIC REASONING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Training multimodal agents to reason over visual observations, executable actions, and environment feedback benefits from grounded trajectories that expose intermediate decisions and link them to verifiable outcomes. Producing such supervision is costly, but each attempt also yields reusable production experience—task constructions, solver outcomes, and artifact-validation evidence—that can support subsequent synthesis. We introduce TRAJFOUNDRY, an experience-driven framework that organizes these results as production experience for subsequent synthesis. These retained outcomes support different parts of subsequent production. *Compositional Task Reuse* preserves executable task families for re-instantiation and structural derivation, including compatible composition; *Validated Skill Reuse* turns valid-task solver outcomes into candidate Skills adopted only after paired Skill replay; and *Targeted Artifact Recovery* uses artifact-validation evidence to repair defective production artifacts at the affected stage while preserving unaffected work. The producer model remains fixed; improvements come from carrying forward reusable task families, validated Skills, and recovery records across production rounds. Under full production-cost accounting, TRAJFOUNDRY produces 591 grounded unique trajectories per 1K cost, compared with 480 for the pre-specified Skill-guided static control and 497 for the adapted open-source EnvScaler baseline. Carry-versus-reset results show gains on newly instantiated tasks alongside recovered prior work. At matched accepted tokens, the TRAJFOUNDRY corpus improves mean structural and environment OOD success by 6.4 points over the Skill-guided static control, and the same corpus advantage transfers to a second student backbone. Our code is available at `https://anonymous.4open.science/r/TrajFoundry-B61F/`

## 1 INTRODUCTION

Multimodal agentic reasoning involves interpreting visual observations, selecting tools and actions, and using environment feedback to guide subsequent decisions (Yao et al., 2022; Schick et al., 2023; Surís et al., 2023; Wang et al., 2025a; Shen et al., 2026). An input–answer pair specifies the task and its final result but does not show which evidence was acquired or how the agent responded to intermediate observations. For supervised training, grounded trajectories make these decisions explicit by pairing executed actions with returned observations and linking the interaction sequence to a verifiable outcome (Hou et al., 2026; Hong et al., 2026; Zhang et al., 2025; Wang et al., 2025a). Producing large, diverse corpora of grounded trajectories requires covering varied combinations of visual reasoning, tool use, and environment interaction, without repeatedly reconstructing the executable tasks and validation procedures behind each new attempt.

Producing these trajectories requires constructing executable tasks, running agents in the corresponding environments, and validating the resulting interactions. Existing systems support this process through environment-grounded task generation (Ramrakhya et al., 2026; Gu et al., 2026), programmatic environment construction and trajectory verification (Song et al., 2026), iterative task expansion or targeted artifact repair (Li et al., 2026; Shi et al., 2026), and memory-based synthesis that carries validation feedback into later generation (Zeng et al., 2026) or connects iterative production to policy improvement (Wang et al., 2025d; Zhang et al., 2026). These advances establish the value of reusing prior production results. The question we study is not simply whether such results should be retained, but which production outcomes can reliably support which subsequent operations.



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.

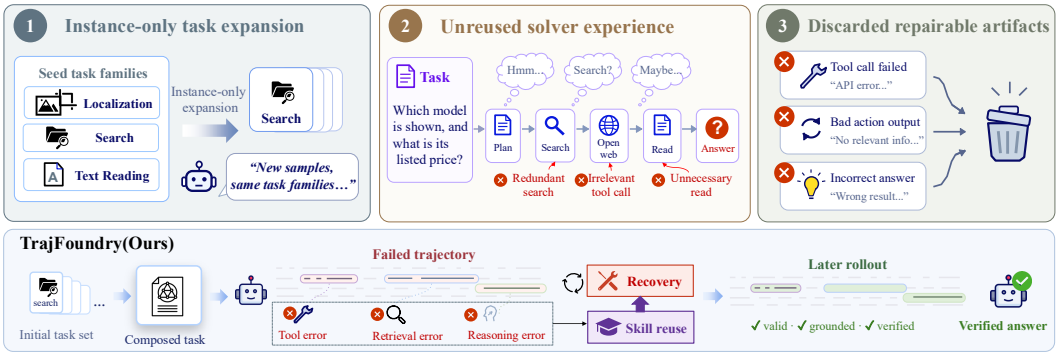


Figure 1: **A running example motivating TRAJFOUNDRY.** Conventional trajectory production can remain within existing task structures, discard valid-task solver experience after individual attempts, and throw away artifacts with repairable construction or grounding defects. TRAJFOUNDRY instead composes reusable executable tasks, validates solver experience as guidance for later rollouts, and routes defective artifacts to the appropriate recovery stage.

Trajectory production leaves task constructions, solver outcomes, and artifact-validation evidence, each of which can support a different part of subsequent production. Figure 1 illustrates how these different results can be underused or discarded in conventional production. We study how to organize them into reliable support for subsequent task construction, solving, and artifact recovery.

We call these accumulated results *production experience*. Carrying this experience forward requires three complementary principles. First, task constructions should remain executable, allowing new instances and structural variations rather than merely preserving completed examples. Second, solver outcomes should become reusable guidance only when they arise from valid tasks and the resulting guidance demonstrates a benefit on subsequent solving. Third, artifact-validation evidence should identify the affected component so that repair preserves unaffected work. These principles require distinguishing solver errors on valid tasks from defects in task construction or execution records: the former can inform solving guidance, whereas the latter require artifact recovery.

We introduce TRAJFOUNDRY, which implements these principles through three connected mechanisms. *Compositional Task Reuse* preserves executable task families for re-instantiation and structural derivation, including compatible composition. *Validated Skill Reuse* turns successes and failures on valid tasks into family-conditioned guidance, adopting a candidate only when paired evaluation against the current guidance supports its utility. *Targeted Artifact Recovery* uses validation evidence to repair affected construction or execution artifacts while preserving unaffected work. The retained task families, validated Skills, and recovery records support later production rounds while all model parameters remain fixed.

We evaluate whether production experience improves subsequent trajectory synthesis and the learning value of the resulting corpus. Under full production-cost accounting, TRAJFOUNDRY produces more verified, non-duplicate trajectories than the compared synthesis baselines. Carry-versus-reset comparisons show that the benefit extends to newly instantiated tasks, rather than arising only from recovered prior work. At matched training tokens, the resulting corpus improves mean structural and environment OOD success by 6.4 points over Skill-guided static production, with the same corpus advantage transferring to a second student backbone. Our contributions are as follows:

- We characterize trajectory synthesis as an experience-accumulating process, distinguishing the training value of accepted trajectories from the reusable production value of task constructions, solver outcomes, and validation evidence.
- We introduce TRAJFOUNDRY, which turns this experience into broader executable task supply, empirically validated solving guidance, and recoverable production work, using validity evidence to distinguish solver errors from artifact defects.
- We show that retained production experience improves synthesis including on newly instantiated tasks, and yields stronger downstream generalization under matched training tokens across two student backbones.

## 2 Related Work

**Executable task synthesis and compositional reuse.** Instruction synthesis expands supervision from seed examples (Wang et al., 2023b; Xu et al., 2023), while agent-oriented systems ground task creation in environment exploration (Ramrakhya et al., 2026), state-based success conditions (Gu et al., 2026), programmatic environment construction with rule-based verification (Song et al., 2026), or recursive expansion of verified tasks (Li et al., 2026; Hua et al., 2026; Guo et al., 2025). Multimodal trajectory datasets record visual evidence, tool actions, and environment observations (Hou et al., 2026; Hong et al., 2026; Zhang et al., 2025; Wang et al., 2025a), but a completed trajectory does not by itself preserve the executable construction and verification procedures needed to instantiate new tasks. Recursive reuse and structural expansion of executable tasks thus have established precedents. TrajFoundry builds on these precedents by treating reusable task constructions as one form of retained production experience, alongside solver outcomes and artifact-validation evidence.

**Trajectory-derived guidance and Skill validation.** Reflection and failure-conditioned learning reuse execution histories (Shinn et al., 2023; Kang et al., 2026; Wang et al., 2026b). Agent Skill Induction compiles successful and failed trajectories into reusable programs (Wang et al., 2025c); Trace2Skill analyzes multiple execution traces and consolidates them into hierarchically structured guidance (Ni et al., 2026); related systems further extend Skill induction through diverse compilation strategies (Shao et al., 2026; Yang et al., 2026; Ma et al., 2026). We use a family-conditioned Skill as production-time guidance, but retain it only after paired evaluation against the current guidance demonstrates improved yield, with induction and validation costs charged to the production budget. This embeds Skill adoption within the production process and subjects it to empirical validation rather than treating compilation as sufficient for reuse.

**Validation-guided recovery and adaptive synthesis.** FACET preserves valid executable state during targeted construction repair (Shi et al., 2026); VISA retains validation evidence and target-model failure patterns to adapt subsequent multimodal data synthesis (Zeng et al., 2026). Both demonstrate that diagnostic information from production can guide repair or adaptation beyond simple accept/reject filtering. Data flywheels (Wang et al., 2025d; Zhang et al., 2026) and co-evolving agents (Lyu et al., 2026; Shukla et al., 2026; Huang et al., 2026) further connect iterative synthesis to model or policy improvement; related systems jointly update task and Skill curricula (Fu et al., 2026; Wang et al., 2023a). Our evaluation fixes producer parameters to isolate the effect of retained production state from parameter learning, and tests whether routing validation outcomes to either valid-task experience reuse or artifact recovery improves subsequent synthesis and downstream generalization under an independent training protocol. Appendix G provides a descriptive comparison with the closest systems.

## 3 Method: TrajFoundry

### 3.1 Production setting and framework overview

TrajFoundry synthesizes a corpus of verified multimodal interaction trajectories within a production budget. It operates in resettable, tool-enabled environments with programmatically checkable outcomes. Production proceeds in rounds, each constructing and executing batches of task instances with a fixed producer $A_{\text{prod}}$. A task instance, or *anchor*, is $x = (I_0, q, E_0, o, \iota_x)$, where $I_0$ is the initial visual observation, $q$ the question or goal, $E_0$ the initial environment snapshot, $o$ a solver-independent programmatic outcome oracle frozen before rollout, and $\iota_x$ a stable instance identifier. The producer receives only the public task, observations, permitted tools, and any validated solving guidance—not the oracle or private environment fields. Its interaction trajectory is

$$\tau = \left(x,\, (r_h, a_h, y_h)_{h=1}^{H},\, \hat{a},\, \mathcal{E}_\tau\right), \tag{1}$$

where $r_h$ and $a_h$ are the reasoning and action at step $h$, $y_h$ the observation returned after executing $a_h$, $H$ the number of interaction steps, $\hat{a}$ the terminal answer, and $\mathcal{E}_\tau$ records evidence for execution validation.

A trajectory record does not itself imply a valid training example. A trajectory enters $\mathcal{D}_t$, the set of trajectories newly accepted in round $t$, only if its task and environment are consistent with the

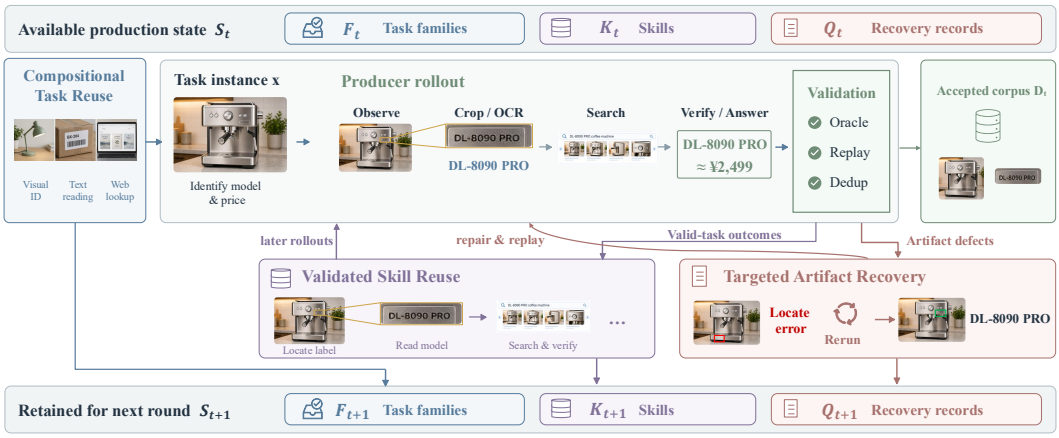


Figure 2: **Experience-driven trajectory synthesis in TRAJFOUNDRY.** Retained production state supports a shared task–rollout–validation spine, where accepted trajectories enter the corpus and valid-task outcomes or artifact defects feed Skill reuse and targeted recovery. The three mechanisms operate around this shared process rather than as serial stages.

oracle, grounding replay from $E_0$ reproduces the observations by re-executing the recorded actions, and the terminal outcome passes $o$. Split constraints and deduplication are applied before inclusion. The aggregate corpus $\mathcal{D}=\bigcup_t \mathcal{D}_t$ is serialized from accepted public interaction fields for downstream training; private validation data and production-time Skill cards are excluded. Production attempts, however, leave more than training examples. Even when a trajectory is not accepted into $\mathcal{D}_t$, the attempt may retain reusable *production experience*: task constructions, solver outcomes on valid tasks, or artifact-validation evidence.

At the start of round $t$, retained experience is organized as $\mathcal{S}_t=(\mathcal{F}_t, \mathcal{K}_t, \mathcal{Q}_t)$, where $\mathcal{F}_t$ stores executable task families together with family-level valid-task outcome summaries used for gap-targeted task discovery, $\mathcal{K}_t$ stores validated family-conditioned solving guidance (Skills), and $\mathcal{Q}_t$ stores recovery records identifying unfinished artifacts and their repair routes. The initial state $\mathcal{S}_0$ contains a calibrated seed-family pool and no Skills or pending repairs. Given fresh seeds $\xi_t$ for task construction and environment instantiation and budget $B_t$, one round returns

$$(\mathcal{D}_t, \mathcal{S}_{t+1}) = \mathrm{Produce}(\mathcal{S}_t, \xi_t, A_{\mathrm{prod}}; B_t). \tag{2}$$

As shown in Figure 2, Compositional Task Reuse supplies the task instances $x$ for Eq. (1) (Section 3.2), Validated Skill Reuse turns valid-task outcomes into guidance for later rollouts (Section 3.3), and Targeted Artifact Recovery returns defective artifacts to the appropriate production step (Section 3.4). They act on a shared production process rather than as three serial stages. Reusable families, Skills, and pending recovery records are drawn from $S_t$; newly committed updates become available through $S_{t+1}$. A newly constructed family may complete its own instantiation, rollout, and validation within round $t$; only its reuse by other production attempts is deferred until commitment to $\mathcal{S}_{t+1}$. Newly adopted Skills similarly guide fresh rollouts only after commitment to $\mathcal{K}_{t+1}$. All model checkpoints remain fixed throughout production (Appendix D.1), and all work—including failed attempts, Skill compilation and evaluation, and recovery—is charged to $B_t$. Downstream training does not feed back into production. Protocol and accounting details appear in Appendix D.

### 3.2 COMPOSITIONAL TASK REUSE

Compositional Task Reuse preserves executable task families rather than only completed examples. A family $f$ retains the procedures needed to generate task inputs, initialize the environment, and construct solver-independent outcome checks, together with compatibility contracts and provenance. Its interface to the production process is

$$x = \mathrm{Instantiate}(f, \xi), \tag{3}$$

where $\xi$ is an instance seed and $\mathrm{Instantiate}$ coordinates task construction, environment initialization, and oracle instantiation to produce the complete task instance $x$ of Section 3.1.

Existing families in $\mathcal{F}_t$ are reused in two ways. *Re-instantiation* preserves the family and its construction and verification procedures while changing the seed. *Structural derivation* changes the task structure and therefore rebuilds the affected construction and outcome-checking procedures. In particular, composition combines parent families only when their registered capability and environment contracts are compatible; it coordinates the parents' construction and environment procedures under a joint task structure and instantiates a new solver-independent oracle for the composed task—answers from parent instances are never reused as the new oracle.

Candidate families are proposed through four registered source routes: persistent capability gaps identified from family-level valid-task outcome summaries (Source A), compatible parent composition (Source B), grounded external intents (Source C), and registered mutation (Source D). Composition and mutation derive new structure from existing families and may draw parents from different source categories (*cross-source* derivation); capability-gap and external-intent routes introduce new target specifications under the same family interface. All routes share the same family interface and validation requirements (Appendix D.2; distinctness uses the identity and deduplication protocol of Appendix E). A candidate is retained only when construction is reproducible, its oracle is stable and solver-independent, its registered contracts are valid and parent-compatible where applicable, and it passes family-level distinctness checks—independently of whether the current producer solves its instances. Retained candidates enter $\mathcal{F}_{t+1}$ for subsequent reuse; construction defects eligible for repair are routed to Targeted Artifact Recovery.

### 3.3 Validated Skill Reuse

A reusable task construction does not by itself provide an effective solving strategy. Validated Skill Reuse supplies structured, family-conditioned natural-language guidance—called a *Skill*—encoding applicability conditions, a solving procedure, and execution checks. A Skill is placed in the producer's rollout context; it neither changes model parameters nor replays a stored trajectory as a demonstration.

For each family, let $K_v$ denote the incumbent guidance: the validated Skill currently stored in $K_t$, or the no-Skill condition before any Skill has been adopted. A complete batch is collected under $K_v$ and outcomes are compiled only from valid task instances with consistent execution records. Verified successes provide effective decisions and checks; solver failures expose residual decision errors. A frozen compiler aggregates the valid-task outcomes collected under $K_v$ into one family-level candidate $K_{\mathrm{cand}}$; private verification fields and instance-specific answers are excluded. Construction and execution-record defects are excluded from Skill evidence and handled by recovery.

$K_{\mathrm{cand}}$ remains a candidate and is not used for subsequent production until validated against the incumbent $K_v$. Paired Skill replay re-solves identical frozen validation instances under $K_{\mathrm{cand}}$ and $K_v$, using the grounding and outcome checks of Section 3.1. Unlike grounding replay, which re-executes a recorded action sequence, it allows different actions while holding the task instance fixed. The validation instances are disjoint from the compilation batch, held-out final evaluation, and the downstream training corpus. A candidate passing the common correctness and grounding checks is adopted only if it improves cost-normalized verified yield after compilation and replay costs are charged and keeps negative transfer—the fraction of paired instances changing from correct under $K_v$ to incorrect under $K_{\mathrm{cand}}$—below the pre-specified tolerance. If adopted, $K_{\mathrm{cand}}$ replaces $K_v$ in $K_{t+1}$ and becomes the incumbent guidance for that family in the next round; otherwise $K_v$ is retained. Appendix D.3 provides the Skill format, compiler prompt, and selection thresholds.

### 3.4 Targeted Artifact Recovery

Skill reuse presupposes valid tasks and consistent execution records. Targeted Artifact Recovery addresses defects in these production artifacts, processing both newly detected defects and unfinished work identified by $\mathcal{Q}_t$. A frozen diagnostic model uses sanitized artifact-validation evidence to select a repair route along the dependency chain. Recovery preserves unaffected validated work and returns the artifact to the affected stage along this chain from family construction to task and environment instantiation to trajectory recording. An invalid specification or compatibility contract requires family repair while retaining valid source and parent structure. An environment–oracle mismatch requires rebuilding the affected instance and refreezing its oracle, while retaining the valid family. An inconsistent execution record requires a new rollout of the verified task. Whenever an upstream component changes, dependent downstream artifacts are regenerated or revalidated.

A faithfully recorded solver failure on a valid task—including a tool error caused by an invalid action chosen by the producer—is instead evidence for Skill reuse, and never justifies modifying that task instance or its oracle. Repaired artifacts return to the corresponding construction, execution, or validation step, and their trajectories enter $\mathcal{D}_t$ only after passing the original corpus checks. Work that remains recoverable but unfinished is retained with its validation evidence and route in $\mathcal{Q}_{t+1}$. Recovery attempts and cross-stage transitions are bounded; Appendices D.1 and D.2 specify the limits and routing contracts.

# 4 EXPERIMENTS

We evaluate whether organizing production experience in this way improves both subsequent trajectory synthesis and the value of the resulting corpus. We first measure end-to-end production efficiency and downstream learning value, then test whether carrying production state improves fresh-seed production, isolate the responsible reuse and recovery paths, and examine whether the gains preserve coverage and diversity.

## 4.1 EXPERIMENTAL SETUP

**Worlds and agents.** We instantiate TRAJFOUNDRY in two controlled, resettable tool-enabled worlds: rendered charts/diagrams and writable visual interfaces, both with deterministic state. A frozen Qwen3.5-9B (Qwen Team, 2026) produces four fresh-seed rounds. A separate copy of Qwen3.5-9B serves as the primary student and is trained with a fixed LoRA-SFT recipe (Hu et al., 2021); InternVL3.5-8B-HF (Wang et al., 2025b) tests transfer to a second learner architecture. Producer-transfer results across 4B–32B open models and a closed multimodal API are reported in Appendix F.5; full controls appear in Appendix D.1.

**Comparisons and budgets.** Baselines include one-shot, independent multi-source, and one-way generate–filter production, EnvScaler (Song et al., 2026) adapted to our common executable substrate, and Skill-guided static production, which retains trajectory guidance while resetting the family pool each round. All methods share the producer, worlds, oracle and acceptance protocol, and normalized production-cost ledger. We use two matched-budget protocols: Protocol B fixes total production cost to evaluate production efficiency, whereas Protocol A matches accepted training tokens after acceptance and deduplication to evaluate downstream data value, with uniform within-family sampling under a 2% cap. Both cover the complete four-round main run and charge all accepted, rejected, replayed, and recovered work. Table 1 reports Protocol-B production metrics together with the corresponding Protocol-A OOD results; Table 2 reports the full Protocol-A downstream evaluation. Protocol-A corpora are generated under their own matched-token production runs rather than by subsampling Protocol-B outputs.

**Metrics and inference.** The primary endpoint is grounded unique trajectories per 1K normalized cost; secondary metrics cover validity, grounding, duplication, coverage, difficulty, recovery, and salvage. Corpus value is evaluated on fresh Family-ID, structural-OOD, environment-OOD, and held-out external tests isolated from production feedback. The external macro averages TIR-Bench (Li et al., 2025) (1,215 items), Agentic-MME (Wei et al., 2026) (418), and VTC-Bench (Zhu et al., 2026) (680, Code-Interpreter track): 2,313 items total. Confirmatory comparisons use paired production seeds; full metric definitions and statistical procedures are given in Appendix E.

## 4.2 MAIN RESULTS: PRODUCTION EFFICIENCY AND LEARNING VALUE

**Production efficiency.** Under equal total production cost, TRAJFOUNDRY yields 591 grounded unique trajectories per 1K cost versus 480 for Skill-guided static production, a paired relative gain of 23.1% (19.4–26.8%; positive in 10/10 seeds). TRAJFOUNDRY also exceeds the adapted open-source EnvScaler baseline ($497 \pm 11$) under the same substrate and cost accounting. The advantage persists under anchor-level deduplication (Appendix E) and across five producers (Appendix F.5).

**Learning value.** Production efficiency does not establish that the resulting data teaches an agent. For the primary student, we train each matched-token corpus with the same LoRA-SFT recipe

Table 1: **End-to-end production efficiency and final-round production diagnostics.** Yield is mean ± standard deviation over ten paired production seeds under iso-production-cost and aggregates the complete four-round run. Quality columns are final-round stage rates rather than four-round aggregate funnel rates. The OOD column reports the corresponding Protocol-A matched-token results.

| Method | Grounded unique / 1K cost ↑ | Valid family rate (%) ↑ | Grounded (%) ↑ | Semantic dup. (%) ↓ | Struct./env. OOD ↑ |
|---|---|---|---|---|---|
| One-shot | 389±10 | 70.8 | 75.4 | 21.3 | 31.8 |
| Independent multi-source union | 414±9 | 72.5 | 78.1 | 19.0 | 35.5 |
| One-way generate–filter | 446±7 | 73.2 | 79.2 | 16.7 | 36.7 |
| EnvScaler-adapted (Song et al., 2026) | 497±11 | 76.5 | 82.7 | 14.8 | 39.7 |
| Skill-guided static production | 480±13 | 73.8 | 84.9 | 15.4 | 39.2 |
| TRAJFOUNDRY | **591±13** | **87.3** | **92.1** | **8.2** | **45.6** |

Table 2: **Fixed-recipe downstream learning at matched accepted tokens.** The base model receives no generated corpus, and the public trajectory corpora are an external data reference rather than a production method. Old regression is the mean accuracy drop, averaged across 148 frozen held-out items that the base student solves correctly (≥50% success over 3 evaluation passes) before any fine-tuning; a positive value indicates forgetting. Trained rows report mean ± standard deviation over three independent optimization seeds.

| Training corpus | Family-ID ↑ | Structural OOD ↑ | Environment OOD ↑ | External macro ↑ | Old regression ↓ |
|---|---|---|---|---|---|
| Base student | 34.8 | 28.0 | 26.4 | 20.0 | 0.0 |
| Public trajectory corpora | 46.5±0.6 | 36.8±0.7 | 34.2±0.7 | 29.8±0.7 | 1.4±0.1 |
| One-shot | 44.2±0.5 | 32.5±0.5 | 31.1±0.8 | 24.0±0.4 | 1.8±0.1 |
| Independent multi-source union | 48.9±0.6 | 36.3±0.4 | 34.7±0.4 | 27.0±0.5 | 1.2±0.1 |
| One-way generate–filter | 51.0±0.9 | 37.4±0.8 | 35.9±0.7 | 27.9±0.8 | 1.3±0.2 |
| EnvScaler-adapted | 51.6±0.7 | 39.6±0.6 | 39.8±0.5 | 29.9±0.6 | 1.2±0.1 |
| Skill-guided static production | 52.8±0.7 | 40.2±0.7 | 38.2±0.5 | 29.4±0.6 | 1.0±0.2 |
| TRAJFOUNDRY | **58.6±0.8** | **46.8±0.7** | **44.4±0.6** | **33.2±0.7** | **0.8±0.1** |

over three optimization seeds; production-time Skills are absent at evaluation. Table 2 shows that TRAJFOUNDRY improves mean structural and environment OOD by 5.9 points over the EnvScaler external baseline and 6.4 points over the substrate-matched Skill-guided static control, while also leading on Family-ID and external evaluation with less old-capability regression. Its external gain is positive on all three constituent benchmarks (Appendix F.2).

Across five production seeds, the structural/environment-OOD advantage over Skill-guided static production remains 6.8 points (95% interval 5.8–7.8; positive in 5/5 seeds). After matching observed depth and capability-cell marginals, the advantage narrows to 3.8 points but remains positive. The same corpus advantage transfers to an InternVL3.5-8B student (Appendix F).

### 4.3 DOES CARRYING PRODUCTION STATE IMPROVE SUBSEQUENT PRODUCTION?

To isolate the effect of retained cross-round production state on subsequent synthesis, we compare carrying $S_t$ with resetting to $S_0$ before each fresh-seed round, while holding acceptance, cost accounting, splits, and deduplication fixed. Because carrying $\mathcal{S}_t$ can improve yield both on fresh anchors and through artifact recovery, we report these contributions separately.

With the producer and per-round budget fixed, carrying $\mathcal{S}_t$ reaches 702 grounded unique trajectories per 1K cost by round four, 57.4% above the reset control (Figure 3(a)). Most of the gain occurs on anchors first instantiated in the current round (659 versus 446 under reset), while recovered prior-round artifacts contribute an additional 43 trajectories (Appendix F.5). Fresh-anchor yield may still include within-round artifact recovery. For clarity, recovered-artifact yield counts only prior-round rejections accepted in a later round, whereas the salvage count in Figure 3(b) also includes

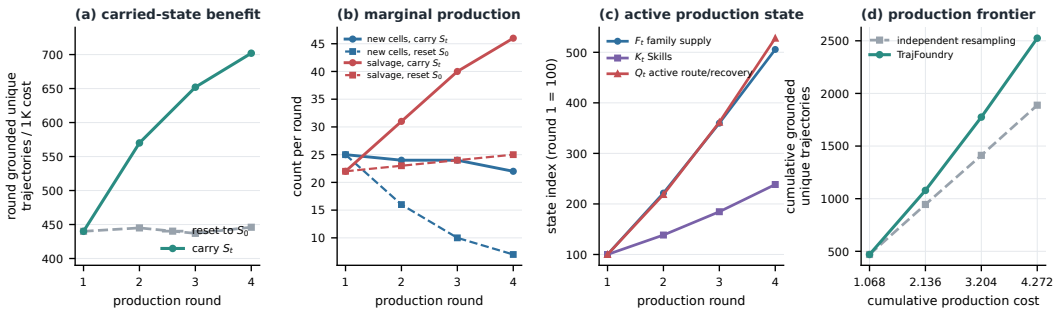


Figure 3: **Subsequent production under carried and reset state.** Curves report means over five paired top-level runs at each round. (a) Total subsequent-round yield, which may include recovered prior work; (b) newly covered capability cells and salvaged artifacts; (c) active production-state components, normalized at round one; and (d) the cumulative production frontier. Audit-only records are excluded from panel (c).

Table 3: **Controlled reuse and recovery mechanisms.** Each row reports a within-panel contrast with non-target mechanisms fixed; absolute values are not compared across panels. Full ladders and configurations are in Appendix F.

| Mechanism | Controlled contrast | Yield / cost ↑ | Mechanism endpoint | Supporting endpoint |
|---|---|---|---|---|
| **Compositional Task Reuse** | Within-source derivation → + cross-source composition | 458 → **474** | Coverage: 127 → **134** cells | OOD: 37.0 → **38.5** |
| **Validated Skill Reuse** | Matched-compute no-Skill → validation-gated replacement | 451 → **492** | Yield/rollout: 408 → **439** | OOD: 36.1 → **38.6** |
| **Targeted Artifact Recovery** | Untyped repair → typed cross-stage routing | 468 → **487** | Salvage: 31.6 → **39.0**% | Grounded: 84.8 → **87.1**% |
| **Path coupling** | Individual paths → both on; fixed family pool | **508** | Yield interaction: +11 | OOD interaction: +0.8 |

within-round repairs. Figure 3(c–d) further shows active-state growth and an outward shift in the cumulative production frontier. The carried-state advantage also persists in a pre-specified eight-round extension, with later-round yield fluctuating around a slower-growth trend (Appendix F.5).

### 4.4 Do the proposed mechanisms work as intended?

To test whether each component improves production through its intended mechanism, Table 3 pairs each functional path with a controlled contrast. Within each panel, starting conditions, budgets, and non-target mechanisms are fixed; the Skill and recovery contrasts additionally use matched anchors.

For Compositional Task Reuse, within-source structural derivation improves yield, while cross-source composition further expands coverage and introduces executable task structures absent from the initial family pool (Appendix F.4). For Validated Skill Reuse, validation-gated replacement outperforms matched-compute no-Skill sampling after accounting for induction and validation costs, while also reducing negative transfer relative to trajectory-local consolidation. For Targeted Artifact Recovery, typed cross-stage routing improves salvage and grounding over both untyped repair and typed local-only routing, supporting validation-guided repair across stage boundaries (Appendix F.4; routing macro-F1: 81.4% in Appendix F.5).

With the family pool frozen, Skill reuse and artifact recovery show a positive interaction: enabling both yields 11 more trajectories per 1K cost than expected from their individual gains alone, with a corresponding +0.8-point interaction on structural/environment OOD. This interaction is larger under growing family supply (Appendix F.4).

Table 4: **Final-round production and corpus diagnostics** relative to Skill-guided static production. Lower is better only for duplicates.

| Metric | Skill-static | TRAJFOUNDRY | Δ |
|---|---|---|---|
| Env.–oracle consistent (%) | 86.8 | **94.7** | +7.9 |
| Grounded trajectory (%) | 84.9 | **92.1** | +7.2 |
| Semantic duplicate (%) ↓ | 15.4 | **8.2** | −7.2 |
| Covered capability cells (/240) | 132 | **164** | +32 |
| Median compositional depth | 2.3 | **2.9** | +0.6 |
| Recovery-rich trajectories (%) | 14.7 | **21.7** | +7.0 |

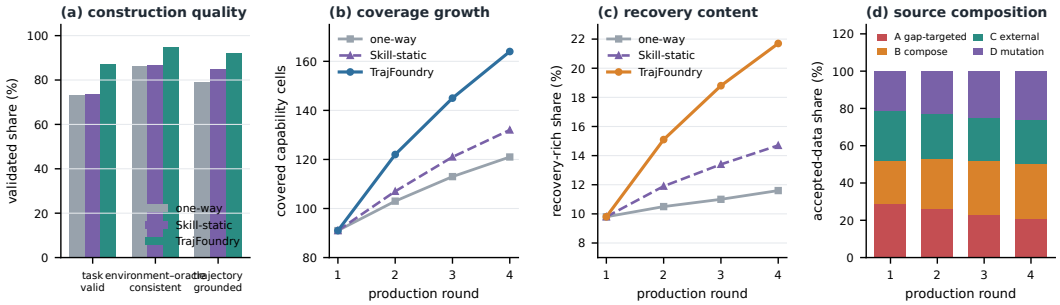


Figure 4: **How the produced corpus changes across rounds.** (a) Final-round construction quality; (b) capability coverage; (c) recovery-rich share; (d) source composition within TRAJFOUNDRY.

### 4.5 CORPUS DIVERSITY AND QUALITY ROBUSTNESS

We next examine whether the production advantage comes from broader, non-redundant data while preserving grounding quality. As summarized in Table 4, TRAJFOUNDRY expands capability coverage from 132 to 164 cells and increases median compositional depth from 2.3 to 2.9, while reducing semantic duplication from 15.4% to 8.2%. Grounding and environment–oracle consistency also improve, indicating that the broader corpus is not obtained by relaxing validation quality. The coverage advantage persists under capability-taxonomy coarsening and matched-count rarefaction (Appendix E, Table 13).

Figure 4 further summarizes final-round construction quality and shows how coverage, recovery-rich share, and source composition evolve across rounds. With proposal quotas fixed, the source mix shifts from gap-targeted and externally grounded tasks toward composition- and mutation-derived families, with composition descendants contributing most newly covered capability cells. TRAJFOUNDRY also contains more recovery-rich trajectories—those in which the solver detects an unreliable intermediate action and corrects its execution before answering. This is a within-trajectory solver behavior, distinct from artifacts recovered by Targeted Artifact Recovery.

## 5 CONCLUSION

We studied how experience accumulated during trajectory synthesis can become a production asset rather than only an accept/reject record. TRAJFOUNDRY preserves executable task families for reuse and structural variation, turns valid-task solver outcomes into Skills that are validated before later use, and uses artifact-validation evidence to recover repairable artifacts while preserving unaffected work. Under full production-cost accounting, this retained production experience improves subsequent production with a fixed producer, while the resulting grounded multimodal trajectories benefit independently trained agents under matched training tokens. Over the measured environments and horizon, the results support structured production experience as a useful resource for subsequent trajectory synthesis.

## AI USE STATEMENT

In this work, we used generative AI tools for task and scene proposal, trajectory generation, failure diagnosis, Skill compilation, and manuscript drafting and editing. All AI-assisted outputs were reviewed and verified by the authors for consistency with the implemented methods and experiments. Generative AI was not used as the outcome oracle or to determine reported evaluation results. We take responsibility for the final content of this work.

## REFERENCES

Tajamul Ashraf, Amal Saqib, Hanan Gani, Muhra AlMahri, Yuhao Li, Noor Ahsan, Umair Nawaz, Jean Lahoud, Hisham Cholakkal, Mubarak Shah, et al. Agent-x: Evaluating deep multimodal reasoning in vision-centric agentic tasks. In *International Conference on Learning Representations*, volume 2026, pp. 49814–49874, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Zenghuang Fu, Zhaoyang Li, Qiuyuan Ai, Haoyu Wu, Minghui Wu, Chenxu Zhao, Ante Wang, Guannan He, and Changwei Wang. Self-play meets skill evolution: Self-evolving search agents that pose, solve, and remember. *arXiv preprint arXiv:2607.29468*, 2026.

Yi Gu, Huacan Wang, Shuo Zhang, Yuqing Hou, Lei Xue, Weipeng Ming, Chen Liu, Fangzhou Yu, Kuan Li, Ronghao Chen, et al. Homeflow: A data flywheel for smart home agent training with verifiable simulation. *arXiv preprint arXiv:2606.01230*, 2026.

Jiacheng Guo, Ling Yang, Peter Chen, Qixin Xiao, Yinjie Wang, Xinzhe Juan, Jiahao Qiu, Ke Shen, and Mengdi Wang. Genenv: Difficulty-aligned co-evolution between llm agents and environment simulators, 2025. URL https://arxiv.org/abs/2512.19682.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, and Guohai Xu. Deepeyesv2: Toward agentic multimodal model. In *International Conference on Learning Representations*, volume 2026, pp. 114851–114872, 2026.

Xinhai Hou, Shaoyuan Xu, Manan Biyani, Moyan Li, Jia Liu, Todd C Hollon, and Bryan Wang. Codev: Code with images for faithful visual reasoning via tool-aware policy optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21500–21510, 2026.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Zhanbo Hua, Yifan Yao, Weihao Xie, Yongchi Zhao, Minghao Liu, Ruizhi Qiu, Zhewei Huang, Zun Wang, Yiyan Ji, Yunhai Ye, et al. Cli-universe: Towards verifiable task synthesis engine for terminal agents. *arXiv preprint arXiv:2606.22883*, 2026.

Siyuan Huang, Pengyu Cheng, Haotian Liu, Tao Chen, Yihao Liu, Jingwei Ni, Shijie Zhou, Ziyi Yang, Gangwei Jiang, Mengyu Zhou, et al. Skill self-play: Pushing the frontier of llm capability with co-evolving skills. *arXiv preprint arXiv:2607.22529*, 2026.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanwei Li, Yu Qi, Xinyan Chen, Liuhui Wang, Jianhan Jin, Claire Guo, Shen Yan, et al. Mme-cot: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency. *arXiv preprint arXiv:2502.09621*, 2025.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Hangoo Kang, Tarun Suresh, Jon Saad-Falcon, and Azalia Mirhoseini. Trace: Capability-targeted agentic training. *arXiv preprint arXiv:2604.05336*, 2026.

Ming Li, Jike Zhong, Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Yuxiang Lai, Chen Wei, Konstantinos Psounis, and Kaipeng Zhang. Tir-bench: A comprehensive benchmark for agentic thinking-with-images reasoning. *arXiv preprint arXiv:2511.01833*, 2025.

Zhongzhi Li, Yucheng Shi, Zongxia Li, Ruhan Wang, Anhao Li, Zixun Huang, Junyao Yang, Lei Ke, Ninghao Liu, Haitao Mi, et al. Recursive synthesis for long-horizon terminal tasks. *arXiv preprint arXiv:2608.05466*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic/skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Yuanjie Lyu, Chengyu Wang, Haonan Zheng, Yuanhao Yue, Junbing Yan, Ming Wang, and Jun Huang. Agenticqwen: Training small agentic language models with dual data flywheels for industrial-scale tool use. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pp. 535–551, 2026.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. Skillgen: Verified inference-time agent skill synthesis. *arXiv preprint arXiv:2605.10999*, 2026.

Zixian Ma, Jianguo Zhang, Zhiwei Liu, Jieyu Zhang, Juntao Tan, Manli Shu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Caiming Xiong, et al. Taco: Learning multi-modal action models with synthetic chains-of-thought-and-action. *arXiv preprint arXiv:2412.05479*, 2024.

Parsa Mazaheri and Kasra Mazaheri. Agentatlas: Beyond outcome leaderboards for llm agents. *arXiv preprint arXiv:2605.20530*, 2026.

Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of symbolic computation*, 60:94–112, 2014.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pp. 17473–17498. PMLR, 2022.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Ram Ramrakhya, Andrew Szot, Omar Attia, Bogdan Mazoure, Anh Nguyen, Yuhao Yang, Zhe Gan, Harsh Agrawal, and Alexander Toshev. Scaling synthetic task generation for agents via exploration. In *International Conference on Learning Representations*, volume 2026, pp. 123630–123660, 2026.

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

Jie-Jing Shao, Haiyan Yin, Yueming Lyu, Xingrui Yu, Lan-Zhe Guo, Ivor Tsang, James Kwok, and Yu-Feng Li. Lifting traces to logic: Programmatic skill induction with neuro-symbolic learning for long-horizon agentic tasks. *arXiv preprint arXiv:2605.01293*, 2026.

Chengyu Shen, Yujie Fu, Gangtao Xin, Yanheng Hou, Wenlong Fei, Guojie Zhu, Jiawei Li, Hongcheng Gao, Runming He, Zhen Hao Wong, et al. Omniabench: Benchmarking general ai agents across diverse scenarios. *arXiv preprint arXiv:2607.14989*, 2026.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Kou Shi, Zun Wang, Qisheng Su, Shiting Huang, Ziao Zhang, Zhen Fang, Qingnan Ren, Jin Liu, Yu Zeng, Yiming Zhao, et al. Facet: Preserving source intent and executable state in terminal task synthesis. *arXiv preprint arXiv:2608.18580*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Aaditya Shukla, Sidney Knowles, Meenakshi Madugula, David Farris, Ryan Angilly, Santiago Pombo, Lu An, Anbang Xu, Abhinav Balasubramanian, Tan Yu, et al. Adaptive data flywheel: Applying mape control loops to ai agent improvement. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 438–454, 2026.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8326–8357, 2026.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11854–11864. IEEE, 2023.

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

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025a.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024a.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, et al. Opencua: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems*, 38:139756–139806, 2026a.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023b.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697, 2024b.

Ziyi Wang, Yuxuan Lu, Yimeng Zhang, Qun Liu, Chen Luo, Jiri Gesi, Hanqing Lu, Yisi Sang, Manling Li, Jing Huang, et al. Sentinel: Failure-driven reinforcement learning for training tool-using language model agents. *arXiv preprint arXiv:2606.12908*, 2026b.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*, 2025c.

Zun Wang, Jialu Li, Yicong Hong, Songze Li, Kunchang Li, Shoubin Yu, Yi Wang, Yu Qiao, Yali Wang, Mohit Bansal, et al. Bootstrapping language-guided navigation learning with self-refining data flywheel. In *International Conference on Learning Representations*, volume 2025, pp. 23542–23568, 2025d.

Qianshan Wei, Yishan Yang, Siyi Wang, Jinglin Chen, Binyu Wang, Jiaming Wang, Shuang Chen, Zechen Li, Yang Shi, Yuqi Tang, et al. Agentic-mme: What agentic capability really brings to multimodal intelligence? *arXiv preprint arXiv:2604.03016*, 2026.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv e-prints*, pp. arXiv–2304, 2023.

Shidong Yang, Ziyu Ma, Tongwen Huang, Xucong Wang, Renda Li, Yiming Hu, Yong Wang, and Xiangxiang Chu. Skillforge: Evolving verifiable skills for reinforcement learning agents. *arXiv preprint arXiv:2608.24747*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Min Zeng, Guanxin Tan, Libin Cen, Yafei Wen, Rui Hu, Liuyang Bian, Xiaolong Chen, and Xiaoxin Chen. Visa: Agentic self-evolving data synthesis for multimodal instruction following. *arXiv preprint arXiv:2608.26013*, 2026.

Kangning Zhang, Weiwen Liu, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weinan Zhang, and Yong Yu. Looptool: Closing the data–training loop for robust llm tool calls. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21144–21164, 2026.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan Fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images, 2025. URL https://arxiv.org/abs/2508.11630.

Hange Zhou, Zhonglin Jiang, Mingzhe Zhang, Xiaotang Wang, Hengwei Dai, Qiyao Yu, Yong Chen, Yongqi Zhang, et al. Clarifyvc: Clarifying ambiguous commands in vehicle control with a hybrid data augmentation pipeline. In *International Conference on Learning Representations*, volume 2026, pp. 122965–122996, 2026a.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *Advances in Neural Information Processing Systems*, 38:113959–113991, 2026b.

Xuanyu Zhu, Yuhao Dong, Rundong Wang, Yang Shi, Zhipeng Wu, Yinlun Peng, YiFan Zhang, Yihang Lou, Yuanxing Zhang, Ziwei Liu, et al. Vtc-bench: Evaluating agentic multimodal models via compositional visual tool chaining. *arXiv preprint arXiv:2603.15030*, 2026.

# A LIMITATIONS

**Scope and verification.** Our claims concern replayable environments whose outcomes admit a deterministic programmatic or tolerance-bounded check. This is what lets every reported oracle be computed from environment state rather than judged by a model, but it also bounds the applicable task space: open-ended or subjective interactions require a different verification model, and we make no claim there. The reported grounding checks establish consistency with replayed actions, observations, and outcomes; they do not establish causal faithfulness of every natural-language reasoning statement. The production advantage also remains positive across the five frozen producers in Appendix F.5, and the data-value ordering holds with a separately initialized InternVL3.5-8B-HF learner (Appendix F.2), but both interaction spaces remain controlled.

**Registered task structure.** Executable-family composition relies on registered capability and compatibility contracts; inducing those contracts automatically is outside the present scope. Routing and semantic-deduplication audits reduce, but cannot eliminate, errors from ambiguous failures or semantically equivalent tasks; the blind audit of Appendix F.5 provides an empirical check on the residual rate at $n{=}100$ per arm rather than eliminating it. The substrate-matched controls compare the implemented reuse and recovery functions under a common interface.

**Horizon, accounting, and Skill scope.** Eight fresh-seed rounds show slower marginal growth over a finite horizon, not an asymptotic trend (Appendix F.5). TaskFamily Skills target family-conditioned production. We do not claim universal cross-family Skill transfer. How Skill benefit changes with producer capability remains open.

# B ETHICS AND RESPONSIBLE RELEASE

The system executes generated actions and may ground public artifacts. Production must therefore run in isolated writable sandboxes with network, filesystem, credential, and action allowlists. Family records retain source, license, and content hashes; personal, confidential, or disallowed content is excluded. External evaluations are isolated from production feedback, and their answers never become production evidence. A release should include accepted and rejected artifacts, route logs, environment images, and known failure modes so users can audit selection bias rather than treating generated trajectories as factual evidence outside their recorded environments.

# C REPRODUCIBILITY STATEMENT

We specify the persistent-state schema, source contracts, evidence routes, split inheritance, matched-budget protocols, and statistical unit in the main paper and Appendix D. The intended release includes immutable environment containers, split manifests, family and Skill lineages, task/trajectory route logs, complete cost ledgers, seed-level metrics, and scripts that regenerate all figures from frozen exports. No trajectory-level observation is used as an independent statistical replicate.

# D METHOD DETAILS

## D.1 GLOBAL PRODUCTION PROTOCOL

The chart–diagram world contains rendered charts, tables, linked notes, and scientific diagrams with crop, OCR, retrieval, and calculation tools. The visual tool-use world contains writable dashboards, documents, and lightweight interfaces backed by deterministic state. Throughout both worlds, the producer is exposed to two tools: a sandboxed **Code** tool, which implements visual manipulation, OCR, calculation, and environment interaction through executable programs, and **WebSearch** for open-web retrieval. Each validation anchor materializes a versioned retrieval snapshot; grounding replay and paired Skill replay both serve WebSearch from this snapshot, so different guidance conditions may issue different queries against the same frozen index without temporal drift. The frozen Qwen3.5-9B producer uses the same tools and decoding protocol in every condition, a 14-turn horizon, and four production rounds with fresh task and scene seeds. Scene rendering is capped at

three attempts, each anchor permits two trajectory regenerations, and a rejected artifact may traverse at most three cross-stage routes. The same frozen Qwen3.5-9B checkpoint serves all model-mediated production roles: trajectory rollout, task and scene proposal generation, failure diagnosis, and Skill compilation. The producer-transfer check replaces only the trajectory-rollout producer with Qwen3-VL-4B-Instruct, InternVL3.5-8B-HF, Qwen3-VL-32B-Instruct, or GPT-5; proposal, diagnosis, and Skill compilation remain fixed to Qwen3.5-9B. Each transferred producer uses the same worlds, tools, action bounds, substrate, and cost ledger with a fixed architecture-compatible serialization. All producer contrasts use ten paired seeds.

**Initialization and verifier boundary.** The round-zero state uses ten seed families retained by a frozen two-sided calibration (producer solves at least one instance; oracle, contracts, and environment pass all acceptance gates) and contains no validated Skill or pending recovery artifact. This solvability check applies only to initial seed selection, not to later retention. In every condition, source operators generate additional families from these seeds within each round; the ten families define the shared starting point, not the complete production pool. Each outcome oracle is computed from deterministic renderer data, sandbox post-conditions, or a declared numeric tolerance and frozen before rollout; no language model generates or adjudicates it. The producer sees only the public task, percept, permitted tools, and any validated Skill; the verifier retains private oracle and environment fields. Skill compilation and recovery receive only sanitized validation outcomes. Student examples are serialized from accepted public interaction records.

A separate copy of the Qwen3.5-9B base checkpoint serves as the primary student; InternVL3.5-8B-HF serves as the second backbone with its own fixed adapter configuration. Within each backbone we fix optimizer, schedule, sequence length, resolution, tool serialization, maximum actions, accepted tokens, and data-order policy across corpora. Production-time Skills are never supplied at evaluation.

The comparison controls share source quotas and the executable substrate with TRAJFOUNDRY. One-shot generates tasks and trajectories independently from the shared seed pool in each matched production round, without carrying state across rounds. Independent multi-source union runs source operators independently and unions accepted trajectories without sharing state across sources or rounds. One-way generate–filter uses the shared substrate and acceptance pipeline but discards rejected artifacts and does not feed production experience back into later synthesis; unused feedback budget is reallocated to independent sampling under the same total cost. EnvScaler-adapted uses the official open-source EnvScaler implementation (Song et al., 2026) adapted to the common executable substrate: SkelBuilder-style programmatic construction, ScenGenerator-style scenario generation, and rule-based trajectory verification are retained, while TRAJFOUNDRY-exclusive mechanisms (carried $F/K/Q$ state, cross-source family composition, trajectory-to-Skill feedback, and typed artifact recovery) are disabled; adaptation details are in Appendix D.4. Skill-guided static production performs complete-batch Skill induction and paired Skill replay on frozen anchors, while disabling cross-round family accumulation, cross-source composition, and construction-artifact recovery; its per-round source operators still generate families from the shared seed pool, but newly retained families do not carry into subsequent rounds. Validated Skills for recurring families remain available across rounds. The mechanism analysis additionally includes the intervention-specific best-of-$n$, trajectory-local consolidation, generic-retry, and untyped-repair controls.

**Matched-budget protocols and endpoints.** Under Protocol A, uniform within-family sampling is applied only after acceptance and deduplication, subject to the 2% per-family cap. Every condition then contributes the same accepted-token budget, but retains the full cost of the corpus produced before sampling. Under Protocol B, total production cost is fixed and the endpoint is the number of validated grounded unique trajectories. Both protocols aggregate a complete four-round run, so early family, Skill, and recovery-state construction is charged to the same denominator as later yield. Downstream optimization compute is held equal and reported separately.

The normalized cost vector is frozen before endpoint evaluation and prices proposal, producer, and verifier tokens; environment steps; Skill induction and paired Skill replay; deduplication; and every rejected attempt. Appendix D.6 specifies the charged resource categories, normalization convention, aggregate main-run cost, and the component-wise cost sensitivity check. The primary endpoint is grounded unique trajectories per 1K normalized cost. Secondary endpoints cover task and environment–oracle validity, grounding replay, semantic duplication, capability coverage, difficulty, recovery-rich share, and salvage.

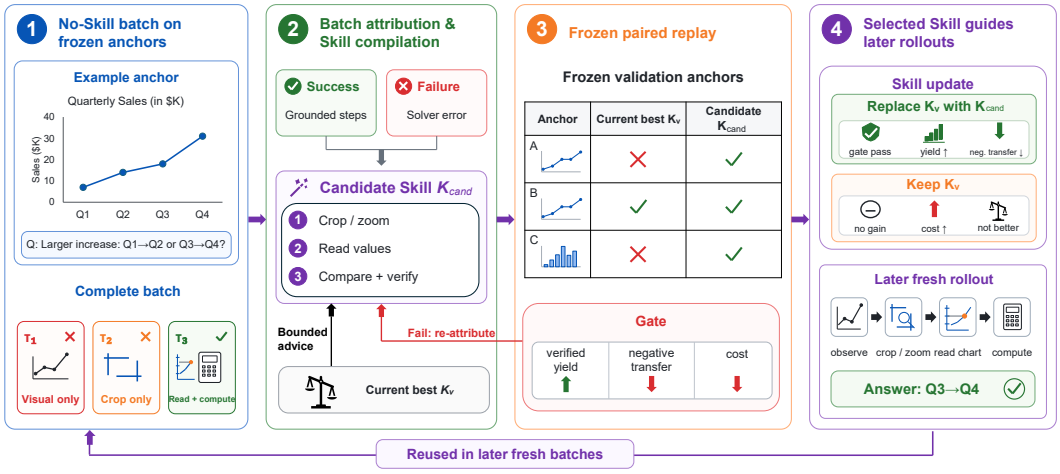


Figure 5: **Validated Skill Reuse.** Valid-task successes and solver failures from a completed batch are aggregated into one candidate Skill; initialization uses a no-Skill batch, while later refinement uses fresh batches under the current validated Skill. Paired Skill replay on frozen anchors compares the candidate with the current-best Skill before the candidate is adopted or rolled back.

### D.2 Task-family construction and routing contracts

Table 5 summarizes the typed contracts. Source operators produce families under the shared interface; validation evidence routes artifacts to recovery or solver outcomes to Skill reuse.

Table 5: Task-supply and evidence-routing contracts. Source operators produce executable families; subsequent evidence either returns invalid construction to Targeted Artifact Recovery or sends valid-task outcomes to Validated Skill Reuse.

| Path | Evidence or source | Operation | Output or destination |
|---|---|---|---|
| Task supply A | Persistent valid-task gap | Isolate or edit a boundary | Target or boundary family |
| Task supply B | Compatible parents | Programmatic composition | Composite family |
| Task supply C | External demand | Ground intent in the sandbox | Demand-grounded family |
| Task supply D | Family or validated Skill | Registered task-space mutation | Neighborhood family |
| Construction | Invalid task or contract | Repair family specification | Rerun family gates |
| Construction | Environment–oracle mismatch | Rebuild environment or oracle | Refreeze environment and oracle |
| Construction | Execution or grounding invalid | Regenerate trajectory | Grounding replay and oracle check |
| Trajectory experience | Valid-task solver outcome | Compile TaskFamily Skill | Paired Skill replay on frozen anchors |

### D.3 Skill induction, selection, and producer integration

Figure 5 gives the complete lifecycle of Validated Skill Reuse, from family-level outcome attribution and candidate compilation to paired replay, replacement, and later reuse. Initialization uses a complete no-Skill batch, whereas subsequent refinement uses fresh outcomes collected under the current validated Skill.

**Producer system prompt.** The producer receives a fixed system prompt adapted from the DeepEyes response protocol (Hong et al., 2026), which defines the action format (<think>, <code>,

<search>, <answer>), the two tool interfaces (a sandboxed Code tool with persistent Python sandbox and a WebSearch tool for open-web retrieval), and reasoning constraints. When a validated Skill exists for the current family, the Skill card is injected between the generic protocol and the runtime reminder; a priority clause then instructs the producer to follow the Skill's graph order and validation rules as the task method. Figure 6 reproduces the effective system prompt (Skill slot marked).

**Validated Skill example.** Figure 7 reproduces validated Skill `sk_panel_diff` for the *before-after panel comparison* family in the visual tool-use world. This family covers any task that presents two corresponding state panels (e.g., shift-change audit sheets, dashboard snapshots, inventory grids) and asks the solver to identify or count the positions that differ. The Skill was compiled with cross-family bounded advice and injected verbatim at the slot shown above.

**Compiler prompt.** The compiler receives the sanitized batch summary with the family contract, verified successes, solver-failure diagnoses, and (when refining) the current validated Skill as bounded advice. Up to two compatible prior Skills, selected by registered family-capability contracts, may also be supplied as compiler-side advice; they cannot override the current family contract or instance-specific evidence. Its fixed prompt is:

```
You are the TaskFamily Skill compiler. Given one family contract and a
construction-valid batch summary, write exactly one complete replacement
TaskFamilySkill. Consolidate verified success strategies and recurring solver
failure fixes into executable steps and evidence checks. Do not invent oracle
values, private environment state, or instance-specific answers. Return
exactly one TaskFamilySkill with fields: family, description, when_to_invoke,
failure_modes, procedure, and validation. Do not append to the existing Skill
or emit alternatives.
```

The compiler produces one complete replacement Skill; instance-specific answers and private oracle values are excluded from its input. A candidate is adopted only if it passes the common correctness and grounding checks, paired negative transfer is at most 5%, and cost-normalized verified yield strictly improves:

$$\frac{\text{verified trajectories under } K_{\text{cand}}}{\text{rollout cost under } K_{\text{cand}} + C_{\text{compile}} + C_{\text{replay}}} > \frac{\text{verified trajectories under } K_v}{\text{rollout cost under } K_v},$$

where $C_{\text{compile}}$ and $C_{\text{replay}}$ are the incremental costs of compiling and evaluating the candidate. Otherwise $K_v$ is retained.

### D.4 EnvScaler baseline adaptation

The EnvScaler baseline (Song et al., 2026) is adapted from the official open-source MIT-licensed implementation. We retain four components from the original framework: (1) SkelBuilder-style programmatic construction of executable task skeletons and environments; (2) ScenGenerator-style scenario generation for task instantiation; (3) rule-based trajectory verification with checkpoint validation; and (4) the iterative environment-scaling loop that expands task coverage through programmatic synthesis. The following adaptations map EnvScaler to our common executable substrate: environment backends are replaced with our chart–diagram and visual tool-use worlds; the producer, tool interface (Code + WebSearch), oracle and acceptance protocol, and normalized cost ledger are shared with all other conditions. TrajFoundry-exclusive mechanisms are disabled: carried $F/K/Q$ production state, cross-source executable family composition, trajectory-to-Skill feedback, and typed artifact recovery are all turned off. EnvScaler retains its rule and checkpoint structure for task-level verification, while instance-level outcome oracles are instantiated by the common deterministic protocol shared across all conditions. The EnvScaler arm therefore tests automatic environment-grounded task construction and rule-based verification without the experience-driven production mechanisms studied in this paper.

### D.5 Splits and evaluation firewall

Each family receives a split signature before any rollout, covering seed ranges, structural templates, asset/domain groups, environment versions, and transitive ancestry. Multi-parent proposals take the

**Figure 6: Producer system prompt (adapted from DeepEyes, abbreviated).** The Skill card is injected at the marked slot; the runtime reminder follows.

````
You are an agent. Keep going until the user's image question is completely resolved.
End only when you have enough real evidence to answer.

Solve the problem step by step. If the answer cannot yet be determined, use the
Code tool to process the image in a persistent sandbox, or use WebSearch for
open-web retrieval. When an injected Skill defines a graph order or shared
representation, that method controls the task strategy.

# response protocol
Every assistant turn must contain a concise reasoning section and exactly one
action, using one of these forms:

<think>what is known, what remains uncertain, and why this action helps</think>
<code>
```python
# one executable Python action
```
</code>
CFY_END_ACTION

or, to retrieve external knowledge:
<think>what factual information is needed and why</think>
<search>concise natural-language query</search>
CFY_END_ACTION

or, when the answer is known:
<think>brief final evidence check</think>
<answer>answer value</answer>

# Code tool
You may run Python in a persistent notebook-style sandbox. The image is at image.png.
Available: numpy, PIL, cv2, scipy, math, re, json.  The sandbox has no outbound
network; use WebSearch for any retrieval need.

# WebSearch tool
Use <search>query</search> to retrieve factual knowledge from the open web. The
returned snippets are shown as the next observation. Use WebSearch when the task
requires external knowledge such as entity identification, factual verification,
or model/domain-specific information that cannot be derived from the image alone.
Do not fabricate factual claims; retrieve and cite instead.

# First-turn visual grounding
The original image is attached to the first user message. From the first turn onward,
use your own vision to identify the target and approximate region. Briefly use that
observation to choose a purposeful first operation. When the Skill requires a full
topology or structured value, establish one full-image coordinate frame or shared
representation named by the Skill.

# Observe, diagnose, repair
Treat every Python output, traceback, and returned image as new evidence. Before the
next block after an error:  Diagnosis: <root cause>; Change: <specific fix>.
Make a minimal executable repair. Never substitute made-up centres, readings, etc.

# Evidence arbitration and stopping
Before another code action, identify the one unresolved question it will answer.
If the action would add no new evidence or merely repeat a measurement, stop.
Before answering, check that the final number agrees with both the semantic visual
interpretation and any targeted tool evidence. Then: <answer>answer value</answer>.

            === [SKILL CARD INJECTED HERE] ===

# Runtime self-check (applies after every Skill above)
Skills above add zero functions to Python. Before emitting code, scan every call.
Each name must be a Python builtin, an imported attribute, or a function defined
above its first call in this block or an earlier successful block.

# Skill-method priority
Use the Skill's graph order, coordinate convention, intermediate representation,
and validation rules as the task method. Complete and validate the next required
graph stage before coding further stages.
````

**Figure 7: Validated TaskFamily Skill** (`sk_panel_diff`, visual tool-use world, compiled with cross-family advice). Injected verbatim into the producer rollout context.

```
TaskFamilySkill  sk_panel_diff
  family: before_after_panel_comparison
  description: Identify or count positions that differ between two
    corresponding state panels in a structured visual interface, using
    a row-major ledger of per-position semantic states.

  when_to_invoke: The image shows two corresponding panels representing
    states at different times or conditions (e.g., Morning vs. Evening
    check, pre- vs. post-update dashboard, before vs. after inventory
    grid), and the question asks which or how many positions changed.

  failure_modes:
    - The two panels cannot be reliably located or paired.
    - Row/column correspondence is ambiguous across panels.
    - Per-position state (color, shape, symbol, value) is uncertain;
      guessing instead of stopping.
    - Decorative elements or grid lines are mistaken for state changes.

  procedure:
    1. Parse the question to identify which two panels to compare and
       what constitutes a "position" (grid cell, slot, indicator, etc.).

    2. Locate both panels in the image. If individual positions are not
       readable at full-image scale, crop or zoom each panel separately.

    3. Build a row-major ledger: for each position (i,j), record the
       state in panel A as S_A(i,j) and in panel B as S_B(i,j), using
       a semantic token (color+shape, text value, or empty).

    4. Compare: mark position (i,j) as changed when S_A(i,j) != S_B(i,j)
       in any semantic dimension. Mark as unchanged only when the full
       state matches exactly, including empty-vs-empty.

    5. Return the result: list of changed positions, or their count,
       as requested by the question.

  validation:
    - The compared panels are the ones specified or implied by the
      question, and the ledger spans every visible position.
    - Comparisons use semantic state, not pixel intensity; borders,
      labels, and decorative marks are excluded.
    - If any position's state is ambiguous after zooming, do not guess.
```

Table 6: Frozen internal evaluation suites. Family counts are unique within a suite; all items are excluded from production and downstream training.

| Suite | Held-out relation | Families | Items |
|---|---|---|---|
| Family-ID | unseen seeds from known families | 64 | 1,201 |
| Structural OOD | held-out composition templates | 27 | 674 |
| Environment OOD | disjoint backend/renderer groups | 19 | 579 |

most restrictive assignment; no OOD family has a parent or sibling in training. External benchmark registries block their hashes, domains, prompts, and answers from proposal evidence. Construction-validation anchors for paired Skill replay are disjoint from final evaluation but may recur across compiler versions.

Table 7 reports a decontamination audit: normalized text overlap, perceptual image hashes, and semantic nearest neighbors are computed between the accepted corpus and every evaluation item. No confirmed overlap was found; all 2,313 items are scored.

Table 7: External-suite decontamination audit. Candidate columns count automatic flags before blind review; "Removed" records confirmed overlaps excluded before scoring.

| Benchmark | Items | Exact | 13-g | 8-g | pHash | Semantic | Removed |
|---|---|---|---|---|---|---|---|
| TIR-Bench | 1,215 | 0 | 2 | 9 | 0 | 6 | 0 |
| Agentic-MME | 418 | 0 | 0 | 3 | 0 | 2 | 0 |
| VTC-Bench | 680 | 0 | 1 | 5 | 0 | 3 | 0 |
| Total | 2,313 | 0 | 3 | 17 | 0 | 11 | 0 |

**External benchmark tool adaptation.** All external evaluations use the same Code+WebSearch interface as production. TIR-Bench is evaluated with executable visual code, VTC-Bench uses its official Code-Interpreter track, and Agentic-MME visual operations are implemented inside the Code sandbox while retrieval is exposed through WebSearch.

### D.6 Production cost, funnel, and bounds

Table 8: Cost items charged before normalization. Attempted and rejected work remain assigned to the condition that produced them.

| Component | Charged event | Raw unit | Attribution |
|---|---|---|---|
| Proposal and materialization | source generation, parsing, rendering, sandbox and oracle construction | tokens / CPU / calls / environment steps | family proposal and task instance |
| Production rollout | actions, observations, retries, and terminal response | tokens / environment steps | trajectory attempt |
| Skill path | attribution, replacement generation, and paired Skill replay | tokens / calls / rollouts | family Skill decision |
| Verification and recovery | task, oracle, grounding, repair, rematerialization, and regeneration | tokens / CPU / environment steps | artifact and route edge |
| Deduplication | structural search and semantic confirmation | CPU / verifier calls | accepted candidate |
| Rejected work | every failed proposal, rollout, or route | corresponding raw units above | originating condition |

Table 8 makes rejected work and the costs of grounding replay and paired Skill replay explicit. Heterogeneous resources are aggregated using a frozen normalization vector estimated from a pre-experiment calibration run that maps each resource to accelerator-equivalent time with model-token cost as the reference unit. The vector is frozen before any endpoint evaluation and is a pre-specified aggregation rule, not a claim about monetary value. Table 9 decomposes the normalized cost by component; Table 10 reports major raw resource-usage statistics in natural units as an auxiliary transparency check.

Table 9: Normalized production-cost decomposition by component, aggregated over the four-round iso-cost run and averaged across ten production seeds. All values are in normalized cost units (K); row totals equal the fixed production budget. Percentages in parentheses are cost shares within each method.

| Method | Model inf. | Env. inter. | Skill/replay | Verif./recov. | Dedup. | Total |
|---|---|---|---|---|---|---|
| One-shot | 3.48 (82%) | 0.50 (12%) | — | 0.22 (5%) | 0.07 (1%) | 4.27 |
| Independent union | 3.33 (78%) | 0.57 (13%) | — | 0.28 (7%) | 0.09 (2%) | 4.27 |
| One-way gen.–filter | 3.08 (72%) | 0.71 (17%) | — | 0.38 (9%) | 0.10 (2%) | 4.27 |
| EnvScaler-adapted | 2.60 (61%) | 1.11 (26%) | — | 0.45 (10%) | 0.11 (3%) | 4.27 |
| Skill-guided static | 2.52 (59%) | 0.82 (19%) | 0.49 (12%) | 0.34 (8%) | 0.10 (2%) | 4.27 |
| TRAJFOUNDRY | 2.38 (56%) | 1.03 (24%) | 0.44 (10%) | 0.34 (8%) | 0.08 (2%) | 4.27 |

Table 10 aggregates raw usage globally by resource type, whereas Table 9 groups the underlying charged events by production function and then reports their normalized cost; because each resource type can contribute to multiple production functions, the two tables are complementary accounting views rather than column-wise conversions.

Table 10: Raw resource usage per method in natural units, aggregated over the four-round run and averaged across ten production seeds. Model tokens are in millions; environment steps, verifier calls, and deduplication checks are in thousands; Skill-compilation calls and replay rollouts are raw counts.

| Method | Model tokens (M) | Env. steps (K) | Skill comp. calls | Replay rollouts | Verifier calls (K) | Dedup. checks (K) |
|---|---|---|---|---|---|---|
| One-shot | 38.4 | 11.2 | 0 | 0 | 3.8 | 2.6 |
| Independent union | 42.1 | 12.8 | 0 | 0 | 4.5 | 2.9 |
| One-way gen.–filter | 47.6 | 14.1 | 0 | 0 | 6.2 | 3.4 |
| EnvScaler-adapted | 51.3 | 16.7 | 0 | 0 | 7.1 | 3.2 |
| Skill-guided static | 48.9 | 14.4 | 384 | 1536 | 6.8 | 3.3 |
| TRAJFOUNDRY | 52.7 | 15.9 | 412 | 1648 | 8.4 | 3.7 |

We do not interpret the normalization as a universal monetary price. To test whether conclusions depend on this particular aggregation, we independently vary every component weight from $0.5\times$ to $2\times$ while holding realized resource usage fixed. The primary TRAJFOUNDRY–Skill-guided-static ordering remains unchanged across all $2^5$ corner combinations of this sweep, indicating that the production advantage is not an artifact of the chosen weights. Accepted-token matching is performed only after corpus acceptance and deduplication.

**Why raw resource totals can differ while normalized costs are equal.** Under Protocol B, total normalized production cost is a *constraint*: every method operates until the same fixed budget is exhausted. The budget is consumed by all charged events—successful rollouts, failed attempts, Skill compilation, verification, recovery, and deduplication—not only by the resources that yield accepted trajectories. A method with higher acceptance rates and fewer rejected attempts converts a larger fraction of its budget into productive work, which can result in higher gross resource usage on accepted events despite the same total cost. Table 10 reports selected major resource dimensions aggregated across all events; it does not include all cost-relevant quantities (e.g., per-event CPU time, materialization and rendering overhead, and call-duration variation), which also carry weight in the normalized cost. Table 9 shows how the fixed budget is allocated across production functions after applying the full normalization vector to the complete per-event ledger. Because the two tables cover different resource scopes and use different aggregation axes, they need not exhibit the same inter-method ordering on every dimension.

**Reproducibility of normalized costs.** The complete per-event cost ledger—recording every charged event with its raw resource consumption, event type, production function, and normalized cost—will be released alongside the codebase and frozen exports (Section C). This ledger enables independent re-derivation of every entry in Tables 9 and 10 as well as the headline yield. The $0.5\times$–$2\times$ sensitivity sweep reported above varies each component weight by a factor far exceeding

any reasonable calibration uncertainty; the primary ordering is unchanged across all $2^5$ weight combinations. We therefore regard the conclusion as robust to the specific normalization, while the released ledger provides full verifiability.

**Production funnel and fixed bounds.** Table 11 summarizes the four-round aggregate funnel for TRAJFOUNDRY. Aggregate rates are lower than the final-round rates in Tables 1 and 4 because early rounds are charged to the same denominator: the valid-family rate rises from 74.2% in round one to 87.3% in round four (81.4% aggregate), and duplicate rate falls from 13.8% to 8.2%. The headline yield comes from aggregate counts: 1,081 retained proposals canonicalize to 527 archive families; 2,525 unique trajectories consume 4.272K normalized cost units, or 591 per 1K cost.

Table 11: Four-round aggregate production funnel, averaged across production seeds. Counts are rounded for display; pass rates are computed from the underlying unrounded aggregate records. These aggregate rates are distinct from the final-round production-stage rates of Tables 1 and 4, which the last column repeats for comparison.

| Stage | Attempted | Passed | Aggr. rate (%) | Final round (%) |
|---|---|---|---|---|
| Proposal retention | 1,328 | 1,081 | 81.4 | 87.3 |
| Environment–oracle materialization | 3,485 | 3,279 | 94.1 | 94.7 |
| Grounding replay | 3,961 | 3,468 | 87.5 | 92.1 |
| Terminal oracle | 3,468 | 2,812 | 81.1 | 85.1 |
| Semantic deduplication | 2,812 | 2,525 | 89.8 | 91.8 |

Aggregated over the run, first-pass grounded-and-oracle success is 80.2% and reaches 85.8% of materialized anchors after bounded retries. The main protocol permits three scene-render attempts, two trajectory regenerations per anchor, three cross-stage route transitions, and 14 dialogue turns; all attempts and terminal dispositions are charged.

### D.7 TRAINING RECIPE

All corpora use the same LoRA-SFT recipe from a shared base checkpoint: adapter configuration, optimizer, schedule, sequence length, resolution, tool serialization, and training steps are fixed across corpora. Families contribute at most 2% of accepted tokens. The primary student is trained under three optimization seeds; the InternVL3.5-8B-HF backbone uses the same frozen corpora with its own adapter configuration. Hyperparameters and checkpoint selection are fixed without consulting external benchmark scores.

Table 12 gives realized trajectory counts and optimization details. Every condition contributes exactly 80.0M accepted tokens (240.0M optimization tokens over three epochs); trajectories are expanded into turn-level examples and packed to 65,536 max context length. Accepted tokens include the full serialized example (prompt, tool-call/observation, and response); loss is computed only on assistant tokens. Skill cards are stripped before student serialization. Both learners use AdamW ($\beta_1$=0.9, $\beta_2$=0.95, weight decay 0.1), bfloat16, unit-norm gradient clipping, and cosine decay with 3% warmup; no evaluation score is used for checkpoint selection.

**Public trajectory reference.** Chosen before downstream evaluation as the closest public sources for the two world interfaces, the downstream reference combines CoTA, the chain-of-thought-and-action corpus released with TACO (Ma et al., 2024), from its curated 293K subset, and AgentNet, the desktop trajectory dataset released with OpenCUA (Wang et al., 2026a), comprising 22.6K human-recorded trajectories. We retain examples with an explicit action–observation sequence, actions mappable to the common student interface, and images compatible with the shared resolution schedule, then apply the same decontamination, token matching, two-world mixture, and LoRA-SFT recipe as for generated corpora. The mapping is necessarily lossy: CoTA provides tool returns but no reinitializable environment, whereas AgentNet pointer and keyboard events are serialized as interface actions. We follow the source licenses, including CoTA's non-commercial research restriction. Without a comparable production cost ledger, these corpora serve only as a downstream data-value reference.

Table 12: Matched-token downstream data and fixed LoRA configurations. Train trajectories are the examples retained after acceptance, deduplication, the 2% family cap, and token matching. Batch is the global number of packed sequences; the final checkpoint at step 936 is used for every evaluation.

| Training corpus | Train trajectories | Accepted tokens | Optimization tokens |
|---|---|---|---|
| One-shot | 2,524 | 80.0M | 240.0M |
| Independent multi-source union | 2,279 | 80.0M | 240.0M |
| One-way generate–filter | 2,222 | 80.0M | 240.0M |
| EnvScaler-adapted | 2,121 | 80.0M | 240.0M |
| Skill-guided static production | 2,139 | 80.0M | 240.0M |
| TRAJFOUNDRY | 2,030 | 80.0M | 240.0M |

| Student checkpoint | LoRA $r/\alpha$/drop. | Target modules | LR | Batch | Steps/epochs | Seq. len. |
|---|---|---|---|---|---|---|
| Qwen3.5-9B | 64/128/0.05 | q/k/v/o + gate/up/down | $2\times10^{-5}$ | 64 | 936/3 | 65,536 |
| InternVL3.5-8B-HF | 64/128/0.05 | q/k/v/o + gate/up/down | $2\times10^{-5}$ | 64 | 936/3 | 65,536 |

## E STATISTICAL ANALYSIS AND STOPPING

**Identity and deduplication.** Family identity is canonicalized by executable semantics and the environment–oracle contract rather than by provenance alone. An anchor identity additionally fixes the family, seed, percept, initial state, question, and frozen oracle; a trajectory identity also fixes the replayed action–observation trace. The 2% corpus cap is applied at family level, whereas the primary production endpoint deduplicates trajectory traces. Corpus acceptance maintains a cumulative identity index over trajectories accepted in previous rounds; this index is shared corpus bookkeeping rather than part of the adaptive production state $\mathcal{S}_t$. We additionally report anchor-level yield as an identity-robustness check.

**Capability taxonomy.** Before inspecting method outcomes, we define a capability cell as the tuple for a frozen anchor $x$, $c(x) = (\mathcal{C}_x, g_x, u_x, e_x)$, where $\mathcal{C}_x$ is the unordered set of required capabilities in the frozen catalog, $g_x$ the composition topology and depth bin, $u_x$ the declared tool pattern, and $e_x$ the world/domain/environment group. Each accepted trajectory inherits its anchor's cell, and coverage is the unweighted number of nonempty cells among grounded unique trajectories. Enumerating registered generators and environment contracts independently of method outputs gives 240 reachable cells in the common substrate; this denominator does not claim coverage beyond the designed executable space.

The statistical unit is one complete top-level production seed; families, instances, and rounds within a seed are not promoted to independent replicates. Production and producer-transfer contrasts use ten paired seeds; downstream robustness uses five production seeds (three optimization seeds for production seed 1). The primary contrast (TRAJFOUNDRY vs. Skill-guided static, iso-production-cost) is frozen before the runs; EnvScaler-adapted is the primary external baseline. Relative yield is $100n^{-1}\sum_s(Y_{\mathrm{ours},s}/Y_{\mathrm{ctrl},s}-1)$ with cluster-bootstrap 95% intervals over paired seeds. A one-sided sign-flip test gives $p = 0.001$ (ten seeds) for the primary contrast and $p = 0.031$ (five seeds, exact floor) for the downstream contrast. Secondary contrasts follow a fixed testing order; once one fails, later $p$-values are descriptive.

Downstream variance is decomposed into production-seed and optimization-seed components. Table 2 reports the mean and standard deviation over three optimization seeds for production seed 1; optimization seeds are paired by identifier across corpora and never pooled with production seeds.

**Identity, matched-count, and taxonomy robustness.** Table 13 collects the three checks on the production and coverage endpoints. Counting unique anchors instead of trajectories keeps the primary advantage; matched-anchor rarefaction separates coverage from corpus size; and two coarser capability taxonomies preserve the same ordering.

Production stops after the fixed four-round main run. An auxiliary saturation extension may stop only after the upper endpoint of the seed-level 95% interval for marginal grounded unique yield falls below

Table 13: Production and coverage robustness. Identity rows report yield per 1K cost. Rarefaction rows are mean ± between-seed standard deviation after within-seed subsampling. Taxonomy denominators appear in parentheses.

| Diagnostic | Setting | One-way | Skill-static | TRAJFOUNDRY | Δ vs. partial |
|---|---|---|---|---|---|
| **(a) Identity-level yield** | | | | | |
| Identity | Unique trajectories | 446 | 480 | **591** | +111 |
| Identity | Unique anchors | 407 | 436 | **533** | +97 |
| **(b) Coverage at matched accepted anchors** | | | | | |
| Rarefaction | 100 anchors | 42±2.5 | 46±2.4 | **62±2.8** | +16 |
| Rarefaction | 250 anchors | 73±3.1 | 82±3.0 | **110±3.4** | +28 |
| Rarefaction | 500 anchors | 101±3.4 | 112±3.2 | **149±3.7** | +37 |
| Rarefaction | 750 anchors | 113±3.0 | 125±2.8 | **161±3.2** | +36 |
| Rarefaction | 1,000 anchors | 119±2.5 | 130±2.3 | **162±2.1** | +32 |
| **(c) Capability-taxonomy granularity** | | | | | |
| Taxonomy | Primary (/240) | 121 | 132 | **164** | +32 |
| Taxonomy | Merged depth bins (/184) | 93 | 102 | **127** | +25 |
| Taxonomy | Tool order removed (/156) | 81 | 89 | **108** | +19 |

one percent of cumulative yield for two consecutive fresh-seed rounds, or after the pre-specified maximum budget. No parametric fit is used to claim unbounded improvement.

# F ADDITIONAL EVIDENCE

## F.1 MAIN-RESULT ROBUSTNESS

Table 14 reports the per-seed production yields underlying the primary contrast in Table 1.

Table 14: Paired production-efficiency yields for the closest partial-loop contrast. Each column is one complete top-level production seed; values are grounded unique trajectories per 1K normalized production cost. Summary statistics in the main text (relative gain, CI) are computed from unrounded seed-level values; displayed yields are rounded for presentation.

| Condition | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Skill-guided static | 485 | 462 | 507 | 489 | 480 | 475 | 488 | 477 | 472 | 465 | 480 |
| TRAJFOUNDRY | 587 | 596 | 573 | 582 | 607 | 601 | 589 | 584 | 613 | 578 | 591 |
| Paired difference | +102 | +134 | +66 | +93 | +127 | +126 | +101 | +107 | +141 | +113 | +111 |

## F.2 DOWNSTREAM ROBUSTNESS

Table 15: Downstream results on corpora from five independent production seeds. Every row uses the same fixed optimization seed (seed 1); for production seed 1, this is one of the three runs summarized in Table 2. The paired difference is TRAJFOUNDRY minus Skill-guided static production on mean structural and environment OOD.

| | One-way generate–filter | | | Skill-guided static | | | TRAJFOUNDRY | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Production seed | Struct. | Env. | Ext. | Struct. | Env. | Ext. | Struct. | Env. | Ext. | Paired diff. |
| 1 | 36.9 | 36.2 | 28.7 | 39.6 | 37.7 | 30.1 | 47.3 | 45.1 | 33.0 | +7.6 |
| 2 | 36.9 | 35.8 | 28.0 | 40.3 | 38.2 | 28.8 | 46.4 | 44.2 | 33.6 | +6.1 |
| 3 | 37.1 | 37.1 | 28.4 | 39.0 | 38.8 | 29.2 | 47.0 | 44.0 | 34.1 | +6.6 |
| 4 | 37.2 | 35.4 | 27.8 | 39.9 | 38.1 | 30.3 | 47.8 | 45.3 | 33.3 | +7.6 |
| 5 | 37.7 | 36.4 | 27.6 | 41.2 | 38.0 | 29.5 | 46.8 | 44.4 | 33.0 | +6.0 |
| Mean | 37.2 | 36.2 | 28.1 | 40.0 | 38.2 | 29.6 | 47.1 | 44.6 | 33.4 | +6.8 |

The paired difference across production seeds has SD 0.8 and 95% interval 5.8–7.8. On the three external benchmarks individually, TRAJFOUNDRY exceeds Skill-guided static by 4.3, 3.5, and 3.7 points, so the macro gain is not driven by one benchmark.

Table 16: Downstream results after resampling each corpus to a common marginal distribution over compositional depth and capability cells. OOD columns are the mean of structural and environment OOD; the delta is TRAJFOUNDRY minus Skill-guided static production on that mean.

| | | Mean struct./env. OOD | | | | External macro | |
|---|---|---|---|---|---|---|---|
| Matching | Kept (%) | One-way | Skill-static | TRAJFOUNDRY | Δ | Skill-static | TRAJFOUNDRY |
| None | 100 | 36.7 | 39.2 | **45.6** | +6.4 | 29.4 | **33.2** |
| Depth matched | 78 | 36.4 | 38.9 | **43.7** | +4.8 | 29.2 | **32.1** |
| Depth and cell matched | 64 | 36.2 | 38.7 | **42.5** | +3.8 | 29.0 | **31.4** |

**Matched task distribution.** Table 16 equalizes depth and capability-cell marginals, discarding 22–36% of tokens. The OOD advantage narrows from 6.4 to 3.8 points but remains positive.

**Second learner.** Table 17 repeats the fixed-corpus comparison with InternVL3.5-8B.

Table 17: InternVL3.5-8B-HF trained on the same frozen, matched-token corpora.

| Corpus | Struct. OOD | Env. OOD | External |
|---|---|---|---|
| One-shot | 29.7 | 28.6 | 21.8 |
| Independent union | 33.2 | 31.4 | 24.1 |
| One-way generate–filter | 34.1 | 32.8 | 25.1 |
| EnvScaler-adapted | 35.8 | 35.3 | 26.8 |
| Skill-guided static | 36.2 | 34.8 | 26.2 |
| TRAJFOUNDRY | **42.4** | **39.6** | **29.8** |

### F.3 SOURCE AND CORPUS ANALYSIS

Table 18 extends the main-text corpus diagnostics (Table 4) to include the one-way baseline. Relative to Skill-guided static production, TRAJFOUNDRY shows higher retention, consistency, and grounding rates, lower semantic duplication, broader capability coverage, deeper compositions, and a larger recovery-rich share. These are final-round diagnostics and are distinct from the four-round aggregate production endpoint in Table 1.

In the final corpus, B descendants account for the largest accepted share (29%) and most newly covered cells (27), whereas A contributes the highest recovery-rich fraction (33.2%).

### F.4 MECHANISM DETAILS

**Structural novelty.** Cross-source composition produces 29.1 new composition topologies absent from $\mathcal{F}_0$, versus 16.7 under within-source, confirming that its coverage gain is not limited to re-instantiating existing structures.

Task-reuse and Skill-reuse panels use half the per-seed production budget with ten paired seeds; recovery and coupling panels use the full per-seed budget with five paired seeds. Skills are disabled in the recovery panel. Under growing supply, the coupling interaction is $+29$ per 1K cost (95% CI $[23.4, 34.6]$; 5/5 seeds), while full-system yield is nearly unchanged (507 vs. 508); the larger interaction reflects lower single-path arms when family growth is charged.

**Skill selection and final-test audit.** Construction-validation anchors select among candidate Skills, whereas the final paired test and later fresh batches are untouched by candidate Skill selection (Table 20). The compiler evaluates 64 families with a median of 2 candidate Skills per family (IQR 1–3, maximum 5). Each candidate is evaluated on 20 frozen construction-validation anchors per family, disjoint from the batch used to compile or refine the candidate, the final paired test, and the

Table 18: **Extended final-round production characteristics** including all three comparison conditions. The compact version appears as Table 4 in the main text.

| Metric | One-way | Skill-static | TRAJFOUNDRY | $\Delta$ vs. partial |
|---|---|---|---|---|
| Valid family rate (%) | 73.2 | 73.8 | **87.3** | +13.5 |
| Environment–oracle consistent (%) | 86.1 | 86.8 | **94.7** | +7.9 |
| Grounded trajectory (%) | 79.2 | 84.9 | **92.1** | +7.2 |
| Semantic duplicate (%) | 16.7 | 15.4 | **8.2** | −7.2 |
| Covered capability cells (/240) | 121 | 132 | **164** | +32 |
| Median compositional depth | 2.1 | 2.3 | **2.9** | +0.6 |
| Recovery-rich trajectories (%) | 11.6 | 14.7 | **21.7** | +7.0 |

downstream training corpus. Under the 5% negative-transfer tolerance, this permits at most one observed correct-to-incorrect change per candidate evaluation.

**Family-level Skill evidence.** In the Skill-reuse mechanism audit, 64 families are evaluated; seven produced a validated Skill that passed the adoption gate. On the disjoint paired test (which does not influence the adoption decision), five of these seven improve, one ties, and one regresses—confirming that the selection gate does not guarantee improvement on every independent held-out set and supporting the conservative adoption threshold.

### F.5 PRODUCTION DYNAMICS AND AUDITS

Table 21 extends the fresh-seed run from four to eight rounds under the stopping rule of Appendix E. The carried-state advantage persists while later-round yield fluctuates around a slower-growth trend. Duplication remains near its round-four level and salvage remains positive. The stopping criterion is not met within the budget, so we make no asymptotic claim.

**Producer transfer.** We repeat the confirmatory Skill-guided-static versus TRAJFOUNDRY contrast with rollout producers spanning two architectures, 4B–32B open checkpoints, and a strong closed multimodal API (proposal/diagnosis/compilation remain Qwen3.5-9B). All five use ten paired seeds. Across all five producers, the paired mean advantage remains positive despite different absolute yields and is positive in 49 of 50 paired runs. This is a focused producer-transfer check rather than a replication of the full experimental suite (Table 22).

**World-stratified routing audit.** The enlarged natural-failure audit samples 240 artifacts from each world. Annotators label the earliest invalid stage or the solver-side boundary without seeing the predicted route; solver-side cases are handed to the Skill path and excluded from non-Skill recovery-effect estimates. Table 23 reports a macro-F1 of 81.4 overall, with lower routing performance in visual tool use than in chart–diagram tasks.

**Independent quality audit of accepted trajectories.** Several quality dimensions—task naturalness, evidence use, action relevance, and semantic non-duplication—are decided by the pipeline's own gates and therefore require independent verification. Two annotators, blind to condition, source route, production round, and pipeline decision, judge 100 accepted trajectories per arm. Samples are stratified by world (50% chart/diagram, 50% visual tool-use) and compositional depth, with a family-level cap to prevent a few high-yield families from dominating. Terminal-oracle correctness is deterministic and is not re-judged.

Table 24 reports five binary criteria: (1) *task naturalness and executability*—the task is coherent, meaningful, and solvable in the presented environment; (2) *evidence consistency*—the trajectory's substantive claims and decisions remain consistent with the observations and tool outputs it actually obtained; (3) *no major irrelevant action*—the trajectory avoids obviously unnecessary, redundant, or off-target actions; (4) *adequate verification*—when ambiguity exists, the trajectory performs sufficient checks before committing to the final answer; (5) *semantic non-duplication*—the trajectory is not a substantive rephrasing of another accepted example (annotators see the nearest-neighbor candidate from automatic dedup). A composite *pass-all* rate summarizes joint quality.

Table 19: Full intervention ladders underlying Table 3. The Skill-reuse ladder charges induction, paired Skill replay, selection, and evaluation to production; the trajectory-local control consolidates the same success and failure evidence into batch-local guidance without cross-batch validation or rollback of candidate Skills; negative transfer is the percentage of anchors across all families in the mechanism audit that a validated Skill turns from correct to incorrect (aggregated over the full disjoint test set, not the per-candidate selection gate). In the recovery ladder, Skills are disabled, valid-task solver failures are excluded, and injected accuracy is the router's stage-label accuracy under deterministic fault injection. For task reuse, *New topo.* is the mean number of distinct accepted composition topologies absent from the shared initial family pool $\mathcal{F}_0$, averaged over the same ten paired production seeds.

**Compositional Task Reuse**

| Condition | Yield/cost | Cells | New topo. | Dup.↓ | OOD |
|---|---|---|---|---|---|
| Instance union | 428 | 106 | 0.0 | 19.6 | 35.4 |
| Private family records | 437 | 114 | 6.3 | 18.2 | 36.0 |
| Shared schema, no transforms | 447 | 120 | 10.4 | 17.1 | 36.5 |
| Executable, within-source only | 458 | 127 | 16.7 | 15.2 | 37.0 |
| Executable + cross-source | **474** | **134** | **29.1** | **13.0** | **38.5** |

**Validated Skill Reuse**

| Condition | Yield/cost | Yield/rollout | Neg. transfer ↓ | OOD |
|---|---|---|---|---|
| No Skill, one rollout | 397 | 397 | 0.0 | 34.6 |
| No Skill, matched best-of-$n$ | 451 | 408 | 0.0 | 36.1 |
| Failure-only compilation | 443 | 415 | 4.8 | 35.7 |
| Success+failure one-shot | 462 | 426 | 4.1 | 36.9 |
| Trajectory-local consolidation | 476 | 434 | 5.1 | 37.7 |
| Iterative replacement | 472 | 432 | 6.6 | 37.1 |
| Validation-gated replacement | **492** | **439** | **2.9** | **38.6** |

**Targeted Artifact Recovery (Skills off)**

| Condition | Injected acc. | Salvage | Yield/cost | Grounded |
|---|---|---|---|---|
| Terminal discard | – | 0.0 | 419 | 78.4 |
| Generic retry | – | 12.8 | 441 | 80.1 |
| Untyped repair | 71.5 | 31.6 | 468 | 84.8 |
| Typed local-only route | 78.6 | 28.4 | 465 | 84.0 |
| Full cross-stage route | **86.3** | **39.0** | **487** | **87.1** |

**Coupling of the two feedback paths, under both supply regimes**

| Condition | Skill feedback | Non-Skill route | Yield/cost | OOD |
|---|---|---|---|---|
| *Frozen family pool* | | | | |
| Neither path | Off | Off | 411 | 35.1 |
| Skill only | On | Off | 448 | 37.2 |
| Routing only | Off | On | 460 | 38.3 |
| Both paths | On | On | **508** | **41.2** |
| Difference-in-differences | | | +11 | +0.8 |
| *Growing family pool* | | | | |
| Neither path | Off | Off | 399 | 34.9 |
| Skill only | On | Off | 433 | 36.6 |
| Routing only | Off | On | 444 | 37.3 |
| Both paths | On | On | **507** | **40.7** |
| Difference-in-differences | | | +29 | +1.7 |

Table 20: Skill benefit across selection and non-selection data. Yield is grounded unique trajectories per 1K total production cost. "Gated" means eligible only through the common corpus acceptance rules.

| Evaluation split | Selects | Corpus | Skill | Matched best-of-$n$ | Gain (%) |
|---|---|---|---|---|---|
| Construction-validation | yes | no | 504 | 451 | 11.8 |
| Final paired test | no | no | 492 | 451 | 9.1 |
| Later fresh production | no | gated | 486 | 449 | 8.2 |

Table 21: Eight-round fresh-seed extension. Rounds 1–4 are the main run. Each entry is the mean over five paired top-level runs. Yield is grounded unique trajectories per 1K normalized cost; cells and salvage are per-round counts; duplicates are the share of candidates rejected at semantic deduplication. Fresh and recovered gains are carried minus reset yield for current-round first-instantiated anchors and prior-round rejections accepted after recovery; they sum to each round's total yield difference.

| Round | Carry | Reset | Fresh gain | Rec. gain | New cells | Carry salv. | Dup. (%) |
|---|---|---|---|---|---|---|---|
| 1 | 440 | 440 | 0 | 0 | 25 | 22 | 13.8 |
| 2 | 570 | 445 | +99 | +26 | 24 | 31 | 10.8 |
| 3 | 652 | 437 | +181 | +34 | 24 | 40 | 9.5 |
| 4 | 702 | 446 | +213 | +43 | 22 | 46 | 8.2 |
| 5 | 728 | 441 | +247 | +40 | 17 | 49 | 8.4 |
| 6 | 751 | 435 | +278 | +38 | 14 | 47 | 7.9 |
| 7 | 746 | 444 | +259 | +43 | 14 | 51 | 8.1 |
| 8 | **773** | 439 | +296 | +38 | 8 | 50 | 7.7 |

Both corpora are largely sound under blind review. TRAJFOUNDRY shows the largest advantage on action relevance (89% vs. 82%), consistent with the role of validated Skill guidance in reducing unnecessary steps. Skill-guided static production scores marginally higher on verification adequacy (87% vs. 86%), plausibly reflecting its simpler task structures. All individual between-corpus margins remain within sampling error at $n$=100 per arm, so the audit serves as an absolute quality check rather than a confirmatory ranking.

## G POSITIONING RELATIVE TO ADJACENT SYSTEMS

Each component in Table 25 has direct precedents; our question is whether structuring them jointly improves later production and downstream learning under one budget.

**Substrate-matched empirical controls.** EnvScaler-adapted uses the official open-source framework (Song et al., 2026) on our substrate with TRAJFOUNDRY-exclusive mechanisms disabled (Appendix D.4). Skill-guided static retains Skill induction and replay while resetting the family pool each round (Wang et al., 2025c). Both controls share the same worlds, seeds, producer, tools, acceptance rules, and cost accounting.

Table 22: Focused producer robustness. Yield is grounded unique trajectories per 1K normalized cost. Gain is the mean paired relative effect and Pos. counts positive paired seeds; all rows use ten paired seeds. The Qwen3.5-9B row reproduces Table 1.

| Producer | Access | Scale | Skill-static | TRAJFOUNDRY | Gain | Pos. |
|---|---|---|---|---|---|---|
| Qwen3-VL-4B-Instruct | Open | 4B | 398 | **479** | +20.4% | 10/10 |
| InternVL3.5-8B-HF | Open | 8B | 441 | **542** | +22.9% | 10/10 |
| Qwen3.5-9B | Open | 9B | 480 | **591** | +23.1% | 10/10 |
| Qwen3-VL-32B-Instruct | Open | 32B | 548 | **631** | +15.1% | 10/10 |
| GPT-5 | Closed | – | 610 | **688** | +12.8% | 9/10 |

Table 23: Natural-failure routing audit enlarged to 480 artifacts and stratified by world. Values are macro precision, recall, and F1.

| World | $n$ | Precision | Recall | F1 |
|---|---|---|---|---|
| Chart–diagram | 240 | 84.2 | 82.0 | 83.1 |
| Visual tool use | 240 | 80.8 | 78.8 | 79.8 |
| Both | 480 | 82.5 | 80.4 | 81.4 |

Table 24: Accepted-corpus quality audit. Two annotators judge 100 accepted trajectories per arm, stratified by world and compositional depth, without access to production condition or pipeline decision. Positive rates are the mean of the two annotators' independent rates; $\kappa$ is computed from the full $2\times2$ agreement table per criterion.

| Criterion | Skill-static | TRAJFOUNDRY | Agree. | $\kappa$ |
|---|---|---|---|---|
| Task natural & executable | 93 | 94 | 96.5 | 0.71 |
| Evidence consistent | 91 | 93 | 95.0 | 0.66 |
| No major irrelevant action | 82 | 89 | 91.0 | 0.64 |
| Adequate verification | 87 | 86 | 92.0 | 0.66 |
| Semantically nonduplicate | 90 | 92 | 94.0 | 0.63 |
| Pass all quality criteria | 73 | 79 | 88.0 | 0.67 |

Table 25: Descriptive positioning relative to adjacent systems. The columns state each work's reported object, retained information, adaptation target, producer-parameter role, and utility evidence; they are not a novelty checklist.

| System | Direct object | Retained information or asset | Used to change | Producer parameters | Subsequent utility evidence |
|---|---|---|---|---|---|
| AutoPlay (Ramrakhya et al., 2026) | Executable task generation | Exploration trajectories and environment evidence | Grounded task proposals | Not the isolating variable | Task-generation and agent evaluations |
| FACET (Shi et al., 2026) | Terminal-task synthesis | Source intent, executable state, and valid components | Targeted construction repair | Not required for repair | Executability and task success |
| RST (Li et al., 2026) | Long-horizon tasks | Verified tasks as recursive seeds | Subsequent task structure | Not required for recursion | Task and agent success |
| Agent Skill Induction (Wang et al., 2025c) | Agent Skills | Successful and failed trajectories compiled as programs | Later task solving | Fixed; Skills guide inference | Held-out task success |
| Trace2Skill (Ni et al., 2026) | Agent Skills | Consolidated trajectory-local lessons | Inference-time task solving | Fixed; Skills guide inference | Skill transfer to later tasks |
| Self-Refining Data Flywheel (Wang et al., 2025d) | Data and navigation policy | Trajectory feedback and refined supervision | Data and learner policy | Updated in the loop | Downstream navigation learning |
| VISA (Zeng et al., 2026) | Multimodal instruction data | Validation evidence and target-model failure memory | Subsequent data synthesis | Fixed; memory conditions generation | Data quality and downstream performance |
| EnvScaler (Song et al., 2026) | Tool-interactive environments | Programmatic task skeletons, scenarios, and rule-based verification | Environment and task coverage via iterative synthesis | Fixed during scaling | Environment diversity and agent evaluation |
| TRAJFOUNDRY | Grounded multimodal trajectories | Executable task families, validated Skills, and artifact-validation evidence | Task supply, rollout guidance, and recovery | Fixed in the main study | Subsequent yield-/cost and an independent fixed student |