ANDES: AGENT-NATIVE DATA EVOLVING SYNTHESIS VIA FEEDBACK-CONTROLLED EXPERIENCE ACQUISITION

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agents are increasingly used to automate LLM post-training, yet acquiring targeted, high-quality training data remains a major bottleneck. Open-web curation requires long-horizon search, filtering, and balancing, while static synthesis pipelines cannot revise their acquisition distributions from intermediate outcomes. To address this gap, we formulate agentic data synthesis as *feedback-controlled experience acquisition*, in which observations from each synthesis round guide where subsequent supervision is acquired. We instantiate this formulation in ANDES (Agent-Native Data Evolving Synthesis). Given a trainer-specified capability target, ANDES maintains an acquisition distribution over a data-synthesis space modeled by a self-evolving World Tree. Diagnostics of effective quantity, topical allocation, and logical diversity then redirect subsequent acquisition toward under-covered capabilities while preserving broad contextual coverage. This synthesis-level feedback enables lightweight adaptation of the training signal before model-level training and evaluation. Across four base models under strict compute constraints, ANDES substantially improves automated post-training, achieves performance comparable to frontier agents on PostTrainBench, and exhibits robust cross-task generalization. Our project is available at `https://anonymous.4open.science/status/ANDES-4DAE`.

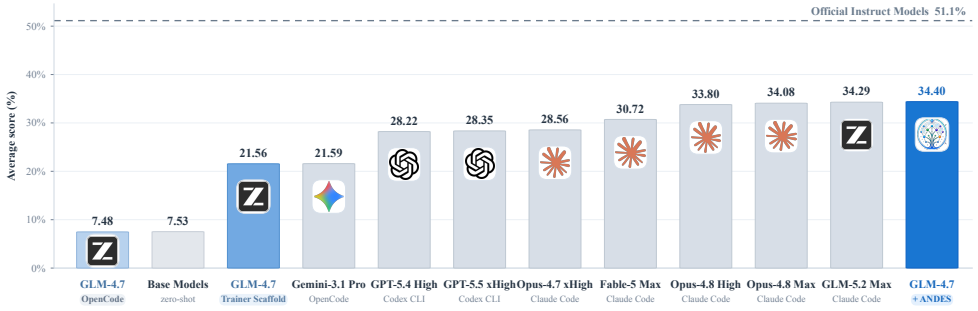


Figure 1: **ANDES brings GLM-4.7 to performance comparable with frontier agents on PostTrainBench.** Its average score increases from 7.48% with the default OpenCode setup to 21.56% after introducing our trainer scaffold, and to 34.40% after further integrating ANDES.

1 INTRODUCTION

Driven by the rapid advancements in agentic capabilities, current research trends are increasingly focusing on applying autonomous agents to orchestrate the Large Language Model (LLM) post-training processRank et al. (2026)—encompassing techniques such as SFT and RLShao et al. (2024); Wu et al. (2026); Yu et al. (2025); Zhao et al. (2026). By delegating the complex decision-making of model alignment to an agent, researchers aim to replace tedious manual trial-and-error with continuous, self-correcting workflowsMa et al. (2026); Jiang et al. (2025); Li et al. (2023; 2026b). In this setting, a post-training agent must diagnose capability deficits and acquire supervision that addresses them. Recent self-improvement systems close feedback loops at the level of optimizer programs, agent implementations, or model parameters by evaluating candidate updates Zelikman

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.

et al. (2024); Zhang et al. (2025); Zweiger et al. (2025). Such feedback becomes available only after a candidate has been executed and evaluated, with weight-level adaptation additionally requiring training. This leaves an important upstream question unresolved: how should an agent construct the training data for a model update?

Addressing this question is difficult because existing data-acquisition strategies fall into two unfavorable regimes. Acquisition from external environments such as the open web requires long-horizon planning, raw HTML parsing, and zero-shot noise filtering Zhou et al. (2024); Feng et al. (2026); the resulting heterogeneous sources must also be organized, balanced, and quality-controlled. Conventional synthetic-data pipelines reduce this environmental burden, but predominantly operate as static, one-shot scripts isolated from the trainer's evolving diagnosis Xu et al. (2025); Liang et al. (2025). Their fixed acquisition distributions can consequently become mismatched to a new target, overproduce redundant supervision, and overfit to benchmark-specific surface patterns. Neither regime gives the trainer a lightweight feedback channel for inspecting the data acquired so far and redirecting subsequent synthesis. Effective autonomous post-training therefore requires the acquisition distribution itself to adapt across synthesis rounds while jointly controlling target relevance, contextual coverage, and sample quality.

To provide this feedback channel, we formulate agentic data synthesis as *feedback-controlled experience acquisition* and instantiate this formulation in ANDES (Agent-Native Data Evolving Synthesis), which adapts synthesis decisions rather than model parameters. Given a trainer-specified capability deficit, ANDES maintains an acquisition distribution over a structured capability space and converts the selected contexts into refined supervision. The synthesis outcome is summarized through diagnostics of effective data quantity, topical allocation, and logical diversity. The trainer can use this report to redirect its next request toward under-covered capabilities, preserve exploration outside dominant regions, and expand repeatedly selected regions instead of recycling saturated contexts. We use the World Tree to model the data-synthesis space as an evolving hierarchy of topics, themes, and scenarios, while report-driven interaction closes the loop between successive synthesis requests. Together, they turn a static generator into a feedback-controlled acquisition process that improves the training signal without requiring an intervening model update.

The main contributions of this work are summarized as follows:

- We formulate agentic data synthesis as feedback-controlled experience acquisition, in which synthesis diagnostics adapt the acquisition distribution across successive requests.
- We introduce ANDES, which realizes this formulation through a structured, self-evolving capability space and report-driven updates that jointly improve target relevance, contextual coverage, and sample quality across acquisition rounds.
- Across four base models and diverse tasks, matched-scaffold ablations, held-out transfer, contamination analysis, internal-model replacement, and repeated runs show that feedback-controlled experience acquisition improves post-training and remains robust across models and tasks.

2 RELATED WORKS

2.1 SFT DATA SYNTHESIS

The growing demand for high-quality supervised fine-tuning (SFT) data has driven extensive research into synthetic data generation. Early works bootstrap instruction data from seed prompts (Self-Instruct Wang et al. (2023), WizardLM Xu et al. (2025), Condor Cao et al. (2025)), but often suffer from limited seed diversity. To reduce explicit seed dependence, Magpie Xu et al. (2024) and MATRIX-Gen Wang et al. (2025) explore template-based and multi-agent generation, though trading flexibility for higher orchestration costs. Another thread focuses on structured synthesis and quality verification via base-refine paradigms (BARE Zhu et al. (2025)), executable feedback (AutoIF Dong et al. (2024)), and prompt decomposition (UltraIF An et al. (2025), DecIF Hui et al. (2025)). More recently, to adapt to evolving model capabilities, Middo Tang et al. (2025) and WIST Li et al. (2026a) introduced dynamic, closed-loop synthesis frameworks. These approaches provide strong generation and verification mechanisms, while leaving limited support for an autonomous trainer to update its acquisition distribution from intermediate synthesis feedback.

2.2 Automated Research and Recursive Self-Improvement

Automating the full research pipeline has emerged as a central challenge in AI development, alongside recursive self-improvement (RSI), where systems revise optimization programs or agent implementations, as explored by STOP and the Darwin Gödel Machine, or generate their own fine-tuning data and update directives, as in SEAL Zelikman et al. (2024); Zhang et al. (2025); Zweiger et al. (2025). Recent works like AIDE Jiang et al. (2025), AlphaEvolve Novikov et al. (2025), AUTO-SOTA Li et al. (2026b), and the AI Scientist series Lu et al. (2024); Yamada et al. (2025) demonstrate the potential of agents to autonomously iterate on experiments and produce algorithmic discoveries. To systematically evaluate these capabilities, benchmarks such as MLE-bench Chan et al. (2025) and MLAgentBench Huang et al. (2024) provide diverse end-to-end ML tasks for agent assessment. Focusing specifically on the post-training phase, TREK Ma et al. (2026) automates the training lifecycle via tree-based exploration, while PostTrainBench Rank et al. (2026) reveals that frontier agents still lag significantly behind official instruction-tuned models under bounded compute budgets. We position ANDES as a complementary mechanism for RSI at the data-synthesis level, using feedback-controlled experience acquisition to translate trainer-specified capability deficits into targeted supervision for subsequent post-training updates.

3 Methodology

3.1 Overview

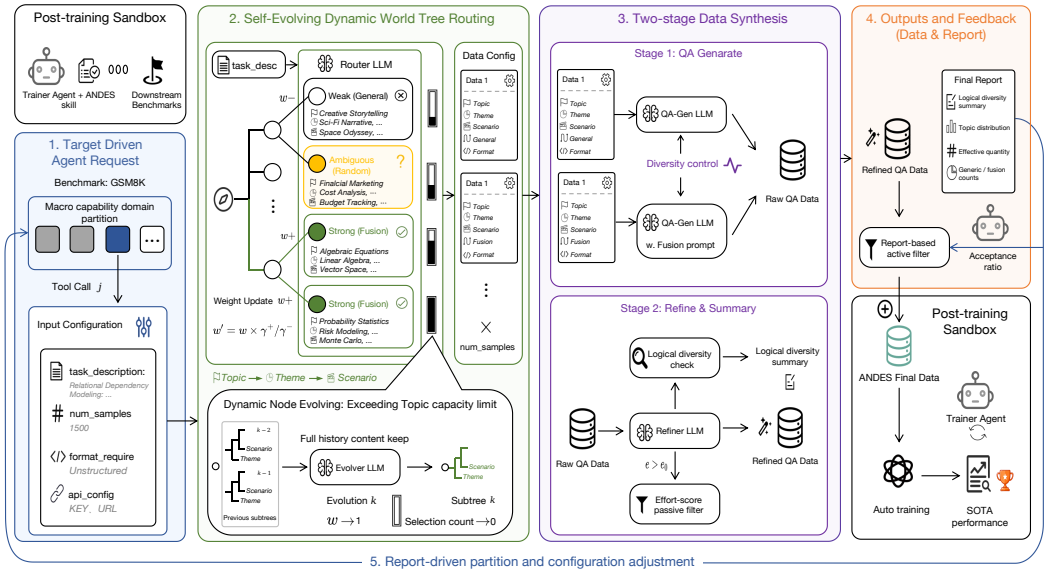


Figure 2: **Overview of ANDES.** A trainer agent maps its post-training objective to capability targets. For each target, ANDES adapts acquisition over a self-evolving World Tree, synthesizes and refines supervision, and returns diagnostics that guide subsequent acquisition decisions.

Operationally, ANDES realizes this formulation by maintaining and updating an acquisition distribution over an evolving space of synthesis contexts. To make this space explicit and extensible, it uses a self-evolving World Tree that organizes candidate contexts hierarchically. A synthesis session begins with a downstream post-training objective, which the trainer abstracts into capability targets. At round j, the trainer specifies a task description τ_j, a requested sample count N_j, and an optional format protocol ϕ_j. Conditioned on this specification, ANDES allocates acquisition over the structured synthesis space, realizes the selected contexts as a refined dataset $\mathcal{D}_j$, and summarizes the observed quantity, coverage, and redundancy in a report $\mathcal{R}_j$. The report then conditions both data retention and the next acquisition decision. The method consequently adapts two coupled objects over synthesis

rounds: the probability assigned to existing regions and the contextual support available within frequently selected regions. As shown in Figure 2, target-driven requests specify the acquisition objective; World Tree routing and evolution adapt its distribution and support; two-stage synthesis realizes selected contexts as supervision; and report-driven adjustment closes the loop across requests.

3.2 TARGET-DRIVEN AGENT REQUEST

Having defined the feedback loop at the session level, we first specify the objective that guides acquisition. Given a downstream post-training objective, the trainer abstracts its benchmark-specific requirements into K macro capability domains $\{d_1, \ldots, d_K\}$. This capability-level representation defines the acquisition targets while reducing dependence on benchmark-specific surface patterns and preserving transferable reasoning requirements across synthesis rounds.

For each domain d_j, the trainer constructs a task description τ_j that characterizes the capability focus, allocates a sample budget N_j, and selects an optional format protocol ϕ_j. The tuple (τ_j, N_j, ϕ_j) specifies the acquisition objective of the j-th ANDES call: τ_j determines the desired capability, N_j controls the amount of supervision, and ϕ_j constrains its representation when a task requires structured outputs. Because later reports can revise these quantities, the decomposition establishes the initial acquisition targets without fixing the full synthesis trajectory in advance.

3.3 SELF-EVOLVING DYNAMIC WORLD TREE ROUTING

Once the capability targets are specified, the next question is how to allocate synthesis toward relevant contexts while retaining broad support. The self-evolving World Tree $\mathcal{W}$ models this support as a Topic $\rightarrow$ Theme $\rightarrow$ Scenario hierarchy (e.g., Cybersecurity $\rightarrow$ Information Leak Prevention $\rightarrow$ Prevention Measures), covering more than 1,000 scenarios at initialization. Let $\mathcal{T} = \{t_1, \ldots, t_M\}$ be its top-level topics. Each topic $t \in \mathcal{T}$ maintains a sampling weight $w_t > 0$ and a selection count c_t, both initialized uniformly at the beginning of a synthesis session. The weight determines how acquisition mass is distributed over the current support, whereas the count records repeated demand for a region and governs when that support should be expanded.

3.3.1 WEIGHTED SAMPLING AND SUITABILITY ROUTING

Within this structured support, suitability routing adapts the probability assigned to existing regions. Given a task description τ and batch size B, the current weights define the topic-level acquisition distribution:

$$t_i \sim \text{Categorical}(w_{t_1}, \ldots, w_{t_M}), \quad i = 1, \ldots, B.$$

For each sampled topic t_i, a theme h_i and a corresponding scenario s_i are drawn uniformly from its subtree. A Router LLM then evaluates all B scenarios in one pass and classifies their suitability for τ as *Strong*, *Ambiguous*, or *Weak*. These judgments divide acquisition into two complementary tracks. *Strong* scenarios enter the target-conditioned *fusion* track, whereas *Weak* scenarios enter the task-agnostic *generic* track. *Ambiguous* scenarios are assigned stochastically to fusion, balancing exploitation of target-relevant contexts with continued contextual coverage. This decision produces a binary fusion flag $f_i \in \{0, 1\}$. Together, the selected path and the trainer specification define the routing configuration

$$r_i = (t_i,\, h_i,\, s_i,\, f_i,\, \tau,\, \phi),$$

which conditions the subsequent synthesis procedure.

The suitability judgments also provide local evidence for updating the acquisition distribution. After routing, topic weights are updated according to

$$w_t \leftarrow \begin{cases} \gamma^+ \cdot w_t & \text{topic } t \text{ has a } \textit{Strong}\text{-rated sample,} \\ \max(\epsilon,\ \gamma^- \cdot w_t) & \text{topic } t \text{ has a } \textit{Weak}\text{-rated sample,} \end{cases} \tag{1}$$

where $\gamma^+ > 1$ increases the mass assigned to regions repeatedly judged relevant, $\gamma^- \in (0, 1)$ discounts weakly related regions, and $\epsilon > 0$ preserves nonzero probability for every topic. The normalized weights therefore act as an online acquisition policy: they progressively concentrate the synthesis budget around contexts aligned with τ while retaining exploration over the rest of the tree.

3.3.2 DYNAMIC NODE EVOLUTION

Reweighting changes how often existing contexts are selected, but sustained concentration can exhaust the finite scenario support of high-relevance topics and induce repetitive training signals. Dynamic node evolution complements reweighting by expanding the support itself. For topic t, an expansion is triggered once its selection count c_t exceeds a saturation threshold proportional to the current size of its scenario pool:

$$c_t > \rho \cdot |\mathcal{S}_t|,$$

where $|\mathcal{S}_t|$ denotes the number of scenarios currently under topic t and $\rho \in (0, 1)$ controls the expansion frequency. Let k denote the current evolution event for t. An Evolver LLM produces a new theme–scenario subtree conditioned on the target and the complete history of earlier expansions for the same topic:

$$\text{Subtree}_k = \text{Evolver}(t,\ \tau,\ \{\text{Subtree}_1, \ldots, \text{Subtree}_{k-1}\}),$$

Access to the expansion history encourages the new subtree to cover contexts not already introduced in earlier rounds. The generated nodes are written back into $\mathcal{W}$, after which c_t is reset to zero and w_t is restored to its initial value so that a newly expanded topic is not immediately over-sampled and expanded again. Routing and evolution thus operate on complementary aspects of the acquisition distribution: routing reallocates probability mass over existing regions, while evolution enlarges the support in regions whose relevance has been repeatedly confirmed.

3.4 TWO-STAGE DATA SYNTHESIS

After routing selects and, when necessary, expands the relevant contextual regions, the selected contexts are converted into reliable training supervision. We separate this realization step into initial QA construction and subsequent quality assessment. Besides improving individual samples, the second stage extracts the data-level observations needed by the feedback loop, linking synthesis quality to later acquisition decisions.

3.4.1 STAGE 1: QA GENERATION

Stage 1 factorizes each routing configuration r_i into question generation and answer generation. The first LLM call synthesizes a question x_i, and the second produces its answer y_i, yielding raw QA data $\{(x_i, y_i)\}$. The fusion flag determines how the capability target enters this construction. When $f_i = 1$, τ is imposed as a hard constraint on the underlying reasoning requirement, while the sampled scenario s_i acts as a contextual wrapper that varies the surface realization. Difficulty level and narrative form are additionally controlled to promote intra-batch diversity. When $f_i = 0$, the prompt conditions only on h_i and s_i, producing general-purpose instruction data in the spirit of knowledge-tree-based synthesis Cao et al. (2025). If a format protocol ϕ is specified, it is attached to the answer-generation prompt for fusion-track samples so that the response follows the required output structure.

3.4.2 STAGE 2: REFINE AND SUMMARY

Stage 2 addresses two complementary failure modes of raw synthesis: insufficient reasoning quality within individual responses and concentration on repeated reasoning templates across the batch. For sample-level quality, a Refiner LLM critiques each raw pair (x_i, y_i) and produces structured feedback covering its strengths, weaknesses, and a revision suggestion. It also assigns an effort score $e_i \in \{1, \ldots, 5\}$ that estimates the cost of repairing the response. Samples with $e_i > e_0$ are removed before rewriting because their defects are unlikely to yield a reliable training signal; the remaining answers are revised according to their critiques, producing refined QA data $\{(x_i, \hat{y}_i)\}$.

For batch-level diversity, we sample several mini-batches from the fusion-track subset of the refined data and ask an LLM to label each as *Low*, *Medium*, or *High* collapse. Here, collapse denotes examples with cosmetically different scenarios but the same underlying reasoning template. We focus this diagnostic on fusion samples because their shared task-specific structure makes them most susceptible to such repetition, while generic-track samples primarily provide broader contextual support. The batch verdicts are aggregated into a natural-language summary of recurring patterns. This summary, together with the topic distribution, effective sample count, and generic/fusion counts,

Table 1: Results on PostTrainBench (accuracy in %), averaged across Qwen3-1.7B, Qwen3-4B, SmolLM3-3B, and Gemma3-4B. The best and second-best agent results among the displayed methods are highlighted in **bold** and underlined; the Official Instruct Models and Base Models rows are references and are excluded from this ranking. Numbers after arrows denote absolute changes over the corresponding zero-shot base-model average, with improvements in red and declines in green.

Method	AIME 2025	ArenaHard	BFCL	GPQA Main	GSM8K	HealthBench	HumanEval	AVG (%)
	(I) Evaluation with Official Instruct Models							
Official Instruct Models	29.17	70.21	85.00	36.21	87.00	43.32	71.49	51.14
	(II) Evaluation with Base Models (Zero-Shot)							
Base Models (Average)	1.67	1.26	1.50	8.48	20.43	9.49	12.81	7.53
	(III) Evaluation with Agent-Post-Trained Base Models							
GPT-5.4 (High)	0.56 (↓1.11)	10.07 (↑8.81)	31.09 (↑29.59)	27.98 (↑19.50)	48.18 (↑27.75)	17.29 (↑7.80)	27.34 (↑14.53)	20.23 (↑12.70)
GPT-5.2	0.83 (↓0.84)	6.61 (↑5.35)	52.50 (↑51.00)	23.72 (↑15.24)	55.90 (↑35.47)	15.81 (↑6.32)	30.23 (↑17.42)	21.38 (↑13.85)
Gemini-3.1-Pro	3.89 (↑2.22)	7.42 (↑6.16)	62.84 (↑61.34)	18.53 (↑10.05)	45.51 (↑25.08)	14.48 (↑4.99)	40.19 (↑27.38)	21.59 (↑14.06)
Opus-4.6	5.00 (↑3.33)	7.78 (↑6.52)	75.92 (↑74.42)	25.52 (↑17.04)	41.04 (↑20.61)	18.81 (↑9.32)	24.75 (↑11.94)	23.16 (↑15.63)
Opus-4.6 (1M)	3.33 (↑1.66)	6.73 (↑5.47)	77.16 (↑75.66)	27.29 (↑18.81)	51.27 (↑30.84)	15.30 (↑5.81)	37.25 (↑24.44)	24.82 (↑17.29)
Opus-4.7 (xHigh)	6.39 (↑4.72)	24.20 (↑22.94)	76.75 (↑75.25)	27.48 (↑19.00)	59.01 (↑38.58)	16.53 (↑7.04)	42.07 (↑29.26)	28.56 (↑21.03)
Opus-4.8 (High)	9.17 (↑7.50)	**31.60** (↑30.34)	72.50 (↑71.00)	21.26 (↑12.78)	69.69 (↑49.26)	35.26 (↑25.77)	**53.35** (↑40.54)	33.80 (↑26.27)
Opus-4.8 (Max)	**10.83** (↑9.16)	31.01 (↑29.75)	83.50 (↑82.00)	23.77 (↑15.29)	68.87 (↑48.44)	**35.43** (↑25.94)	40.39 (↑27.58)	34.08 (↑26.55)
GLM-5.2 (Max)	7.78 (↑6.11)	29.32 (↑28.06)	80.83 (↑79.33)	**28.93** (↑20.45)	72.71 (↑52.28)	28.41 (↑18.92)	50.05 (↑37.24)	34.29 (↑26.76)
	(IV) Evaluation with Proposed Method (Compared with Baseline)							
GLM-4.7 (OpenCode)	1.67 (↑0.00)	1.26 (↑0.00)	1.50 (↑0.00)	8.48 (↑0.00)	18.76 (↓1.67)	9.49 (↑0.00)	13.88 (↑1.07)	7.48 (↓0.05)
GLM-4.7 & ANDES (Ours)	3.90 (↑2.23)	26.98 (↑25.72)	**86.85** (↑85.35)	28.43 (↑19.95)	**73.35** (↑52.92)	32.78 (↑23.29)	50.04 (↑37.23)	**34.40** (↑26.87)

* Model not submitted – base model score shown † Evaluation error – base model score shown

forms the synthesis report $\mathcal{R}_j$. Thus, Stage 2 produces both the refined supervision and an observable description of how the current acquisition decision was realized.

3.5 Report-Driven Partition and Configuration Adjustment

The preceding stages produce both refined supervision and a diagnostic account of the realized acquisition outcome. The report $\mathcal{R}_j$ closes the loop through two coupled updates. First, it conditions which part of the current batch enters the training pool. The trainer applies report-conditioned filtering to $\mathcal{D}_j$, discards a report-dependent proportion of samples, and retains $\mathcal{D}_j^{\text{keep}} \subseteq \mathcal{D}_j$. The accepted data are accumulated as $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_j^{\text{keep}}$. This update links the observed quality and redundancy of the current synthesis round to the effective amount of supervision used for training.

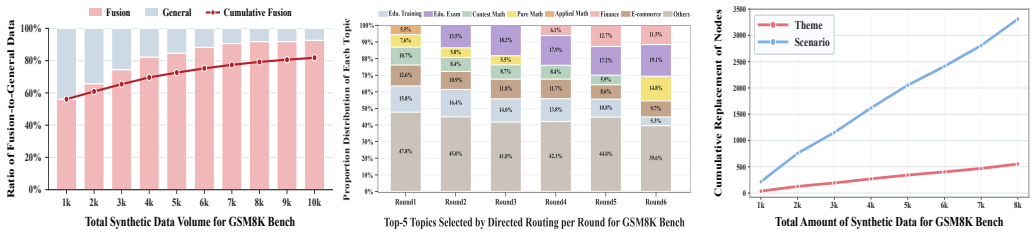
Second, the same report conditions where and how the next batch is acquired. The collapse-pattern summary guides the trainer to revise τ_{j+1} toward underexplored capability angles, while the sample deficit $N_j - |\mathcal{D}_j^{\text{keep}}|$ informs an upward adjustment of N_{j+1} to compensate for discarded samples. The format protocol can likewise be revised when the report indicates a mismatch in output structure. The next configuration is therefore determined by the current synthesis outcome:

$$(\tau_{j+1},\, N_{j+1},\, \phi_{j+1}) = \mathcal{F}\Big(\tau_j,\, \phi_j,\, \mathcal{R}_j,\, N_j - |\mathcal{D}_j^{\text{keep}}|\Big)\,,$$

where $\mathcal{F}$ denotes the trainer's report-conditioned update policy. The resulting loop couples selection into the accumulated training pool with reallocation over the synthesis space: routing determines where experience is acquired, refinement and diversity analysis reveal how that decision materializes in data, and the report converts these observations into the next request. This feedback-controlled experience-acquisition process progressively improves target relevance, contextual coverage, and data quality across synthesis rounds without requiring an intervening model update.

4 Experiments

We organize the evaluation around four research questions. RQ1 examines whether feedback-controlled experience acquisition improves post-training across base models and target tasks. RQ2 studies how the acquisition distribution adapts through directed routing and node evolution. RQ3



(a) General-to-Fusion Data Ratio across Different Data Scales

(b) Directed Routing Performance on Top-5 Topics under Interaction Mechanism

(c) Continuous Evolution of Knowledge Tree Nodes under Increasing Data Scales

Figure 3: **Visualization of ANDES routing and node evolution mechanism.** (a) The increasing fusion-data ratio shows that routing gradually shifts toward GSM8K-relevant nodes.(b) Top-routed GSM8K topics contain higher fusion ratios, indicating effective allocation to target-aligned capability regions.(c) The growing number of evolved themes and scenarios shows that ANDES refreshes frequently selected nodes to preserve data diversity.

isolates the sources of improvement under matched training conditions, and RQ4 examines whether targeted acquisition preserves broad contextual coverage.

4.1 EXPERIMENTAL SETUP

Trainer Agent Scaffold To ensure the proper execution of basic training procedures, we construct a lightweight, CLI-based trainer scaffold, which denotes the training-execution framework evaluated without ANDES in our scaffold comparison. We provide it with a fundamental scaffold necessary to execute training workflows, encompassing essential tools (e.g., `bash`, `read`, `write`), sub-agents such as a training monitor, and a skill system designed for the integration of ANDES. To ensure fairness, we evaluated the performance of this scaffold independently in Section 4.4. We select GLM-4.7 as the driving model, as its moderate agentic capabilities provide an ideal comparative baseline to demonstrate our method's efficacy.

Evaluation Our experiments are primarily conducted on PostTrainBenchRank et al. (2026). Following its official protocols, we deploy a sandbox environment on a single GPU with a 10-hour wall-clock time limit. We evaluate all four officially supported base models (Qwen3-1.7B, Qwen3-4B, SmolLM3-3B, and Gemma-3-4B) across seven benchmarks, with four independent runs for each setting. These span mathematical reasoning (GSM8K Cobbe et al. (2021), AIME 2025), code generation (HumanEval Chen et al. (2021)), tool use (BFCL Patil et al. (2025)), scientific QA (GPQA-Main Rein et al. (2023)), medical dialogue (HealthBench Arora et al. (2025)), and open-ended writing (ArenaHardWriting Li* et al. (2024)). For every (model, benchmark) pair, we run the trainer agent end-to-end and report the official metric of the corresponding task. Full implementation and evaluation settings are detailed in Appendix A. Additional multi-target, data-quality, contamination, and held-out-transfer analyses are reported in Appendix B.

4.2 RQ1: DOES FEEDBACK-CONTROLLED EXPERIENCE ACQUISITION IMPROVE POST-TRAINING?

Autonomous Post-Training on PostTrainBench. The results on PostTrainBench are reported in Tab. 1 and Fig. 4. Across four base models and seven benchmarks, ANDES reaches an average score of 34.40%, placing it alongside the strongest agents in the comparison, including GLM-5.2 (Max) at 34.29%, Opus-4.8 (Max) at 34.08%, and Opus-4.8 (High) at 33.80%. ANDES also obtains 86.85% on BFCL and 73.35% on GSM8K. Under the same trainer scaffold, adding ANDES raises the aggregate score from 21.56% to 34.40%, an absolute gain of 12.84 points. Improvements appear across tool use, mathematical reasoning, and open-ended writing and remain consistent across the four base-model settings. GLM-4.7 with the default OpenCode setup obtains 7.48%, close to the 7.53% zero-shot average, while the matched scaffold comparison separates gains from training execution and feedback-controlled experience acquisition. Detailed per-base-model results and four-run statistics are reported in Appendix C.

4.3 RQ2: How Does the Acquisition Distribution Adapt?

Analysis of Directed Routing and Node Evolution. We examine whether the acquisition distribution becomes increasingly target-relevant while retaining broad contextual support during a GSM8K synthesis session (Fig. 3). Rather than uniformly sampling the World Tree, the directed routing mechanism actively shifts its focus according to the target task. As synthesis progresses, the system increasingly allocates its budget to target-relevant nodes, driving a steady rise in the proportion of "fusion data" while general data diminishes (Fig. 3a). This indicates that routing becomes progressively more selective across synthesis rounds. A concrete breakdown of this behavior (Fig. 3b) shows that ANDES homes in on high-value regions, such as mathematical reasoning and quantitative relations. By constraining generation within these localized topics, the framework injects explicit reasoning logic while preserving diverse contextual backgrounds, combining task-specific supervision with broad scenario coverage.

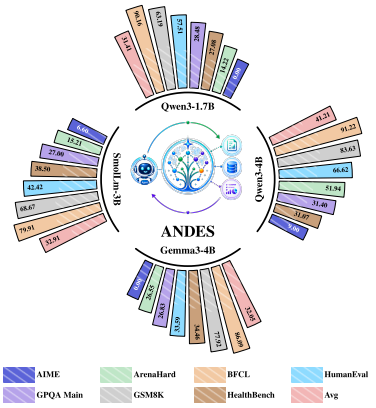


Figure 4: **PostTrainBench results across four base models.** Colors denote the seven tasks and their weighted average.

Crucially, this concentrated routing does not lead to diversity collapse, a common pitfall when specific scenarios are oversampled. The node evolution mechanism counteracts this risk by continuously refreshing high-relevance regions of the World Tree. As the GSM8K-oriented data volume scales, the cumulative number of newly evolved themes and scenarios keeps increasing rather than hitting a ceiling (Fig. 3c). Whenever highly relevant nodes are repeatedly queried, ANDES spawns novel sub-scenarios instead of recycling exhausted templates, maintaining fresh yet task-relevant training contexts across synthesis rounds. Together, directed routing ensures cognitive alignment with the target task, while node evolution provides a continual supply of contextual diversity. As a result, ANDES can continuously generate vertical training data without sacrificing diversity, supporting robust and scalable autonomous alignment.

4.4 RQ3: What Drives the Improvement?

Decoupling the Source of Improvements. Fig. 5 decouples the performance gains achieved by our proposed framework. We evaluate four distinct settings across the four base models (Qwen3-1.7B, Qwen3-4B, SmolLM3-3B, and Gemma-3-4B) separately: the original base model, GLM-4.7 with the default OpenCode setup, GLM-4.7 with our trainer scaffold but without ANDES, and the full GLM-4.7+ANDES system.

Replacing the generic OpenCode baseline with our trainer scaffold yields substantial initial gains by stabilizing data formatting, execution, and training orchestration. Under this matched scaffold, adding ANDES produces a further 12.84-point aggregate improvement, with gains across BFCL (59.25 to 86.85), GSM8K (47.24 to 73.35), and HealthBench (9.68 to 32.78). This progression separates the contribution of reliable training execution from that of feedback-controlled experience acquisition. When averaged across four base models, ANDES improves every evaluated task over the scaffold-only condition. The weighted-average gains range from 11.96 to 14.31 points (Appendix C), indicating that the benefit is not driven by a single model family.

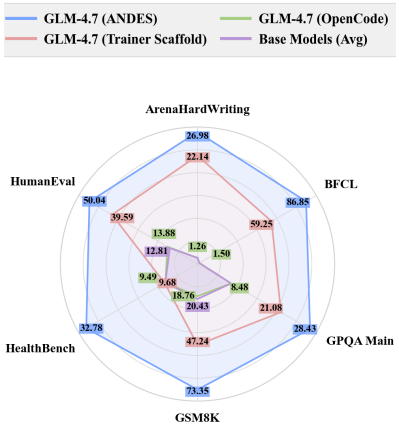


Figure 5: **Matched-scaffold comparison across four base models.** ANDES improves the average by 12.84 points.

Ablation on System Components. Tab. 2 compares the principal system variants under the same Qwen3-1.7B trainer scaffold and training setup. GLM-4.7 with OpenCode obtains 7.11 average

Table 2: Experiments results on PostTrainBench (accuracy in %) based on the Qwen3-1.7B base model. The best and second-best results among the ANDES variants are highlighted in **bold** and underlined. Numbers after arrows denote absolute changes over the corresponding zero-shot base model, with improvements in red and declines in green.

Method	AIME 2025	ArenaHard	BFCL	GPQA Main	GSM8K	HealthBench	HumanEval	AVG (%)
Qwen3-1.7B(Base)	0.00	0.91	0.00	14.06	12.66	7.54	7.93	6.66
GLM-4.7 (OpenCode)	0.00 (↑0.00)	0.91 (↑0.00)	0.00 (↑0.00)	14.06 (↑0.00)	12.66 (↑0.00)	7.54 (↑0.00)	12.20 (↑4.27)	7.11 (↑0.45)
ANDES w/o World Tree	**0.00**	4.92	70.00	22.22	49.00	26.72	54.71	25.97
ANDES w/o Interaction	**0.00**	8.02	68.00	25.41	59.86	22.61	50.01	26.58
ANDES w/ Gemini Backend	**0.00**	13.84	**92.00**	24.37	**64.37**	**27.67**	**57.83**	30.84
ANDES w GLM-4.7 (Ours)	**0.00** (↑0.00)	**14.22** (↑13.31)	90.16 (↑90.16)	**28.48** (↑14.42)	63.19 (↑50.53)	27.08 (↑19.54)	57.51 (↑49.58)	**31.41** (↑24.75)

performance. Removing the World Tree and using a Condor-style general-purpose synthesis path Cao et al. (2025) yields 25.97; without dynamic routing and expansion over capability domains, this variant has weaker target-aware domain coverage. Removing report-driven interaction yields 26.58, whereas full ANDES with GPT-4o as the backend for all internal LLM modules reaches 31.41. Replacing all internal LLM modules with Gemini-3.1-Flash produces 30.84, 0.57 points below the default system, showing that ANDES remains stable across different model backends. The default system is strongest on ArenaHard (14.22), GPQA Main (28.48), and weighted average (31.41), while the Gemini-backed system is strongest on BFCL, GSM8K, HealthBench, and HumanEval. Together, these results support the contribution of the World Tree and interaction loop as well as the robustness of the overall system.

4.5 RQ4: DOES THE WORLD TREE PRESERVE BROAD COVERAGE?

Breadth and Universality of The World Tree. Beyond directed routing and node evolution, the effectiveness of ANDES also relies on the broad synthesis space provided by its World Tree. The World Tree covers diverse macro-level knowledge domains and organizes them into a hierarchical topic-theme-scenario structure, containing 72 topics, 394 themes, and 1,182 scenarios. This large-scale hierarchy prevents data synthesis from being restricted to a single task domain or a fixed set of templates. As a result, ANDES can preserve general contextual coverage while routing synthesis toward task-relevant nodes for vertical capability improvement. The feature-space visualization further shows that synthetic samples for different benchmarks form distinguishable task-related regions while still sharing overlapping areas, indicating that the World Tree supports both target alignment and cross-task generality. We provide a more detailed analysis of the World Tree coverage, hierarchical composition, and feature-space distribution in Appendix D. The embedding geometry provides two complementary signals about the acquisition distribution. Separation among task-conditioned regions reflects specialization toward distinct capability targets, while their shared support reflects contexts that remain useful across tasks. Together with the held-out transfer results, this pattern supports an acquisition process that increases target relevance without collapsing onto benchmark-specific surface forms.

5 CONCLUSION

We introduced ANDES, an agent-native data synthesis framework that formulates data generation as feedback-controlled experience acquisition. Given trainer-specified capability targets, ANDES maintains an acquisition distribution over a structured synthesis space modeled by a self-evolving World Tree. Directed routing concentrates acquisition on relevant capability regions, node evolution expands repeatedly selected regions to sustain contextual coverage, and two-stage synthesis produces refined supervision together with diagnostics that guide subsequent acquisition decisions. Under the same trainer scaffold, adding ANDES improves the PostTrainBench aggregate from 21.56% to 34.40%, achieving performance comparable to frontier agents. Component ablations, replacement of all internal LLM modules, held-out transfer, contamination analysis, and four-run results further support the effectiveness, robustness, and generalization of feedback-controlled experience acquisition. These findings highlight adaptive control of the data-acquisition process as an effective approach to autonomous LLM post-training.

REFERENCES

Kaikai An, Li Sheng, Ganqu Cui, Shuzheng Si, Ning Ding, Yu Cheng, and Baobao Chang. Ultraif: Advancing instruction following from the wild, 2025. URL https://arxiv.org/abs/2502.04153.

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. Healthbench: Evaluating large language models towards improved human health, 2025. URL https://arxiv.org/abs/2505.08775.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Maosong Cao, Taolin Zhang, Mo Li, Chuyu Zhang, Yunxin Liu, Haodong Duan, Songyang Zhang, and Kai Chen. Condor: Enhance llm alignment with knowledge-driven data synthesis and refinement, 2025. URL https://arxiv.org/abs/2501.12273.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. Mlebench: Evaluating machine learning agents on machine learning engineering, 2025. URL https://arxiv.org/abs/2410.07095.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models, 2024. URL https://arxiv.org/abs/2406.13542.

Yukang Feng, Jianwen Sun, Zelai Yang, Jiaxin Ai, Chuanhao Li, Zizhen Li, Fanrui Zhang, Kang He, Rui Ma, Jifan Lin, Jie Sun, Yang Xiao, Sizhuo Zhou, Wenxiao Wu, Yiming Liu, Pengfei Liu, Yu Qiao, Shenglin Zhang, and Kaipeng Zhang. Longcli-bench: A preliminary benchmark and study for long-horizon agentic programming in command-line interfaces, 2026. URL https://arxiv.org/abs/2602.14337.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation, 2024. URL https://arxiv.org/abs/2310.03302.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*, 2023.

Tingfeng Hui, Pengyu Zhu, Bowen Ping, Ling Tang, Guanting Dong, Yaqi Zhang, and Sen Su. Decif: Improving instruction-following through meta-decomposition, 2025. URL https://arxiv.org/abs/2505.13990.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code, 2025. URL https://arxiv.org/abs/2502.13138.

Fangyuan Li, Pengfei Li, Shijie Wang, Junqi Gao, Jianxing Liu, Biqing Qi, and Yuqiang Li. Wist: Web-grounded iterative self-play tree for domain-targeted reasoning improvement, 2026a. URL https://arxiv.org/abs/2603.22352.

Haoyuan Li, Hao Jiang, Tianke Zhang, Zhelun Yu, Aoxiong Yin, Hao Cheng, Siming Fu, Yuhao Zhang, and Wanggui He. Traineragent: Customizable and efficient model training through llm-powered multi-agent system, 2023. URL https://arxiv.org/abs/2311.06622.

Jijie Li, Li Du, Hanyu Zhao, Bo wen Zhang, Liangdong Wang, Boyan Gao, Guang Liu, and Yonghua Lin. Infinity instruct: Scaling instruction selection and synthesis to enhance language models, 2025. URL https://arxiv.org/abs/2506.11116.

Tianle Li*, Wei-Lin Chiang*, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024. URL https://lmsys.org/blog/2024-04-19-arena-hard/.

Yu Li, Chenyang Shao, Xinyang Liu, Ruotong Zhao, Peijie Liu, Hongyuan Su, Zhibin Chen, Qinglong Yang, Anjie Xu, Yi Fang, Qingbin Zeng, Tianxing Li, Jingbo Xu, Fengli Xu, Yong Li, and Tie-Yan Liu. Autosota: An end-to-end automated research system for state-of-the-art ai model discovery, 2026b. URL https://arxiv.org/abs/2604.05550.

Hao Liang, Xiaochen Ma, Zhou Liu, Zhen Hao Wong, Zhengyang Zhao, Zimo Meng, Runming He, Chengyu Shen, Qifeng Cai, Zhaoyang Han, Meiyi Qiang, Yalin Feng, Tianyi Bai, Zewei Pan, Ziyi Guo, Yizhen Jiang, Jingwen Deng, Qijie You, Peichao Lai, Tianyu Guo, Chi Hsu Tsai, Hengyi Feng, Rui Hu, Wenkai Yu, Junbo Niu, Bohan Zeng, Ruichuan An, Lu Ma, Jihao Huang, Yaowei Zheng, Conghui He, Linpeng Tang, Bin Cui, Weinan E, and Wentao Zhang. Dataflow: An llm-driven framework for unified data preparation and workflow automation in the era of data-centric ai, 2025. URL https://arxiv.org/abs/2512.16676.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Zerun Ma, Guoqiang Wang, Xinchen Xie, Yicheng Chen, He Du, Bowen Li, Yanan Sun, Wenran Liu, Kai Chen, and Yining Li. Trex: Automating llm fine-tuning via agent-driven tree-based exploration, 2026. URL https://arxiv.org/abs/2604.14116.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. Posttrainbench: Can llm agents automate llm post-training? 2026. URL https://arxiv.org/abs/2603.08640.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Zinan Tang, Xin Gao, Qizhi Pei, Zhuoshi Pan, Mengzhang Cai, Jiang Wu, Conghui He, and Lijun Wu. Middo: Model-informed dynamic data optimization for enhanced llm fine-tuning via closed-loop learning, 2025. URL https://arxiv.org/abs/2508.21589.

Dong Wang, Yang Li, Ansong Ni, Ching-Feng Yeh, Youssef Emad, Xinjie Lei, Liam Robbins, Karthik Padthe, Hu Xu, Xian Li, Asli Celikyilmaz, Ramya Raghavendra, Lifei Huang, Carole-Jean Wu, and Shang-Wen Li. Matrix: Peer-to-peer multi-agent synthetic data generation framework, 2025. URL https://arxiv.org/abs/2511.21686.

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, 2023. URL https://arxiv.org/abs/2212.10560.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of sft: A reinforcement learning perspective with reward rectification, 2026. URL https://arxiv.org/abs/2508.05629.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions, 2025. URL https://arxiv.org/abs/2304.12244.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing, 2024. URL https://arxiv.org/abs/2406.08464.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (stop): Recursively self-improving code generation, 2024. URL https://arxiv.org/abs/2310.02304.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

Zhengyang Zhao, Lu Ma, Yizhen Jiang, Xiaochen Ma, Zimo Meng, Chengyu Shen, Lexiang Tang, Haoze Sun, Peng Pei, and Wentao Zhang. Gift: Reconciling post-training objectives via finite-temperature gibbs initialization, 2026. URL https://arxiv.org/abs/2601.09233.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

Alan Zhu, Parth Asawa, Jared Quincy Davis, Lingjiao Chen, Boris Hanin, Ion Stoica, Joseph E. Gonzalez, and Matei Zaharia. Bare: Leveraging base language models for few-shot synthetic data generation, 2025. URL https://arxiv.org/abs/2502.01697.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models, 2025. URL https://arxiv.org/abs/2506.10943.

APPENDIX CONTENT

A Implementation Details **14**
A.1 Sandbox, Hardware and Evaluation Protocol 14
A.2 Trainer Agent Scaffold 14
A.3 ANDES Internals 15
B Generalization and Reliability Analyses **16**
B.1 Multi-Target Generalization 16
B.2 Data Quality Validation 17
B.3 Contamination Analysis and Held-Out Generalization 17
C Detailed PostTrainBench Results **19**
C.1 Stability Across Independent Runs 19
C.2 Results Across Base Models 19
D World Tree Coverage **22**
E Case Study: Agent Trajectory **22**

A IMPLEMENTATION DETAILS

This section documents the evaluation environment, trainer scaffold, and synthesis parameters required to reproduce our experiments.

A.1 SANDBOX, HARDWARE AND EVALUATION PROTOCOL

To ensure rigorous and reproducible evaluations, all PostTrainBench experiments are conducted within an isolated, standardized sandbox environment. We abstract away low-level dependency management by exposing unified interfaces to the trainer agent, allowing it to focus entirely on optimization strategies rather than system-level troubleshooting. Each post-training session is strictly constrained to a 10-hour compute budget on a single A100 GPU, enforcing realistic resource bounds for autonomous alignment. Upon timeout, only the designated final checkpoint is extracted for assessment. To strictly prevent train-test leakage or environmental manipulation, downstream evaluations are executed in an independent, pristine sandbox. Furthermore, in accordance with the PostTrainBench protocol, an independent auditor module (powered by GPT-5.1) monitors the agent's trajectory to detect and prevent reward-hacking behaviors, such as evaluation tampering or unauthorized model substitution, thereby guaranteeing the integrity of the final results.

All reported PostTrainBench results follow the same task suite, scoring rules, and sandbox configuration. Controlled comparisons within this paper are matched in trainer scaffold, compute environment, data budget, and training configuration.

A.2 TRAINER AGENT SCAFFOLD

Tool action space. The trainer agent operates over a fixed set of primitives, summarized in Table A.1. `bash` runs in two modes: foreground commands time out after 1 800 s with a hint to retry in background mode, while background commands are launched via `nohup`, registered in an internal job manager, and observed through the `wait` and `read_file` tools. `search_dataset` wraps the HuggingFace Hub API and is reserved for the *data_loader* sub-agent.

Sub-agents. Two sub-agents are pre-registered. The *data_loader* (tools: `search_dataset`, `read_file`, `write_file`, `bash`, `Skill`) is responsible for dataset discovery, sampling-based audit and Alpaca-format normalization, and is the entry point through which ANDES is invoked. The *training_monitor* (tools: `bash`, `read_file`, `wait`, `Skill`) supervises a launched training job,

Table A.1: Tool action space exposed to the trainer agent.

Tool	Purpose
`bash`	Shell execution (foreground / background modes)
`read_file`	Range or tail reads on workspace files (15 KB output cap)
`write_file` / `edit`	Workspace I/O and exact-match edits
`search_dataset`	HuggingFace Hub dataset retrieval
`wait`	Synchronization for long-running jobs (capped at 3 600 s)
`todo`	Multi-step plan tracking
`task`	Dispatch a sub-agent (`data_loader` or `training_monitor`)
`Skill`	Load a skill file into the conversation

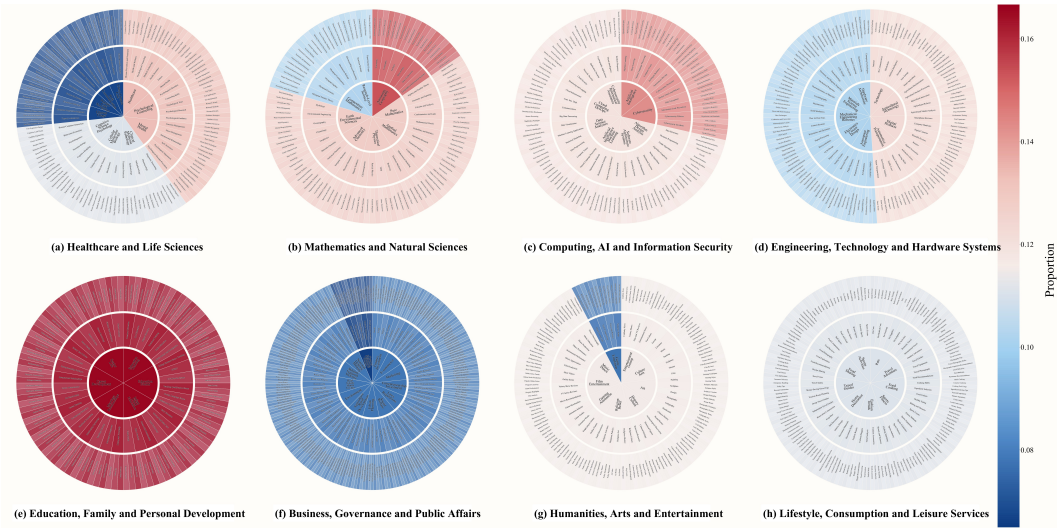


Figure A.1: **Visualization of the knowledge coverage of the ANDES world tree.** It shows the hierarchical distribution of all world-tree nodes across broad macro domains, covering diverse topics, themes, and scenarios.

parses ETA from the training log, and verifies that artifacts exist before reporting back. Both sub-agents communicate with the trainer agent only via a file-based prompt, ensuring a clean separation of context.

Skills. Capability-specific guidance is delivered through three Markdown skill files loaded on demand: `posttrain skill` (engineering principles for the full training lifecycle), `llamafactory skill` (dataset registration, YAML template, training settings), and `data-strategy skill` (the ANDES skill: domain decomposition, report-driven discard ratio decisions, format-protocol selection rules, and seven failure-signal taxonomy). Note that we provide only basic llamafactory configuration options to ensure that training proceeds normally, thereby isolating the contribution of ANDES). The ANDES skill enforces (i) macro-level domain decomposition into $\sim$6 cognitive-operation domains, (ii) one ANDES call per domain with 1000–2000 raw samples and discard ratio $\in \{0\%, 10\%, 20\%\}$ chosen per round from the report.

A.3 ANDES INTERNALS

We build our code upon DataFlowLiang et al. (2025) framework.

Internal LLMs. The router, generator, refiner, evolver, and diversity summarizer are all instantiated with gpt-4o at temperature 0, sharing a single API gateway. In the Gemini-backend ablation, all of these internal LLM modules are replaced with Gemini-3.1-Flash.

World tree. The initial taxonomy contains 72 topics, 394 themes, and 1 182 scenarios, grouped into 8 macro categories (Healthcare and Life Sciences; Mathematics and Natural Sciences; Computing, AI and Information Security; Engineering, Technology and Hardware Systems; Education, Family and Personal Development; Business, Governance and Public Affairs; Humanities, Arts and Entertainment; Lifestyle, Consumption and Leisure Services).

Routing parameters. Topics are weighted-sampled in mini-batches of $B = 5$, with all topic weights initialized to 1.0. After each batch is rated by the router, the reward factor for *Strong*-rated topics is $\gamma^+ = 1.5$ and the decay factor for *Weak*-rated topics is $\gamma^- = 0.8$, clipped at a floor $\epsilon = 0.1$. *Ambiguous* samples are assigned to the fusion track with probability 0.8. Each ANDES call routes $\lceil N/3 \rceil$ scenarios for N requested samples.

Subtree evolution. A topic triggers expansion once its cumulative selection count exceeds $\rho = 0.8$ times the current size of its scenario pool. The evolver is prompted with the topic, the task description, and the full history of previously generated subtrees, and returns 6 new themes, each containing 6 scenarios (i.e., 36 new scenarios per evolution). After expansion, the topic's selection count and weight are reset to their initial values to prevent cascaded re-evolution.

Refinement. Stage 2 first asks the refiner to critique each QA pair and emit an effort score $e_i \in \{1, \ldots, 5\}$. Samples with $e_i > 4$ are dropped before any rewriting; the rest are rewritten conditioned on the original pair and the critique. The diversity check then samples mini-batches of 25 items from the fusion-track subset, targeting $\sim$20% coverage of the fusion pool, and aggregates per-batch *Low/Medium/High* verdicts plus a free-form pattern summary into the report returned to the trainer agent.

Format protocols. The trainer agent selects `format_requirement` once per benchmark from {`unstructured`, `code`, `tool_call`}; the protocol is attached only to the answer prompt of fusion-track samples and constrains the output to contain at least one fenced code block of the corresponding type. `format_requirement` is fixed across rounds within the same benchmark.

B Generalization and Reliability Analyses

We examine whether the observed gains persist beyond the primary benchmark setting through multi-target transfer, deterministic data-quality checks, contamination analysis, and held-out evaluation.

B.1 Multi-Target Generalization

While the main text demonstrates the efficacy of ANDES in the standard autonomous PostTrainBench setting (single benchmark per training), a truly robust data synthesis framework must be capable of generalizing across diverse tasks without suffering from capability degradation. In this section, we extend our evaluation to validate the cross-task generalization and multi-target synthesis capabilities of ANDES. Specifically, we prompt the trainer agent to utilize ANDES to simultaneously acquire a balanced dataset that covers all target benchmarks.

Experimental Setup. We employ the trainer agent to simultaneously synthesize data targeting five benchmarks (AIME24, GaoKaoZhong et al. (2023), MBPPAustin et al. (2021), MMLUHendrycks et al. (2021), CEVALHuang et al. (2023)) in a single session. We then evaluate and report the SFT performance on the Qwen3-8B model, comparing our results against other high-quality dataset baselines: (1) InfininstructLi et al. (2025), a large-scale dataset containing millions of high-quality instruction-tuning data. (2) DataflowLiang et al. (2025), a static two-stage generic data synthesis pipeline. We synthesis 10,000 data using GLM-4.7+ANDES, and sampled 10,000 training data points from the baseline datasets, trained for one epoch with learning rate 2e-5. To further compare data efficiency, we also compared the training results using 1 million data from the Infininstruct dataset.

Results and Discussion. As shown in Table B.1, ANDES significantly outperforms both the 10K-sampled baselines and the 1M full-scale static dataset across the diverse evaluation suite. Notably, conventional static pipelines and large-scale data mixtures (such as Infininstruct and Dataflow) often

Table B.1: Results across five diverse benchmarks. The best and second-best results are highlighted in **bold** and underlined, respectively. Numbers after arrows denote absolute changes over the corresponding base model with improvements in red and declines in green.

Method	Data Volume	AIME24	Gaokao	MBPP	MMLU	CEval	Overall
Base Model(Qwen3-8B-Base)	–	13.3	35.2	82.5	73.0	82.5	57.3
+ Infinstruct	10k	0.0 (↓13.3)	34.1 (↓1.1)	76.2 (↓6.3)	74.2 (↑1.2)	81.3 (↓1.2)	53.2 (↓4.1)
+ Infinstruct	1M	0.0 (↓13.3)	28.6 (↓6.6)	70.1 (↓12.4)	71.3 (↓1.7)	77.9 (↓4.6)	49.6 (↓7.7)
+ Dataflow	10k	**10.0** (↓3.3)	31.9 (↓3.3)	77.2 (↓5.3)	74.6 (↑1.6)	82.1 (↓0.4)	55.2 (↓2.1)
+ **ANDES (Ours)**	10k	**10.0** (↓3.3)	**49.5** (↑14.3)	**77.7** (↓4.8)	**74.8** (↑1.8)	**82.4** (↓0.1)	**58.9** (↑1.6)

struggle with capability degradation during multi-target SFT, leading to noticeable performance drops compared to the base model. In contrast, ANDES not only effectively mitigates these severe regressions but also drives exceptional gains in specific domains. Particularly striking is its performance on the Gaokao benchmark, achieving a substantial absolute improvement of 14.3 points while all other static baselines suffered severe degradation. This remarkable contrast underscores the critical necessity of actively acquiring targeted, in-domain data, demonstrating that ANDES's dynamic routing and scenario abstraction can seamlessly adapt to localized knowledge requirements to yield a comprehensively balanced curriculum for robust cross-task generalization.

B.2 Data Quality Validation

We complement model-based refinement with deterministic validators on subsets of the training data synthesized by ANDES for downstream target tasks that admit symbolic, executable, or schema-based checks. The validation results are summarized in Table B.2.

Table B.2: Deterministic validation of training data synthesized by ANDES, grouped by the downstream task used to guide synthesis. Each check applies only to the indicated subset of synthetic samples.

Synthesis target	Validator	Evaluated subset	Result
GSM8K	SymPy equation equivalence	3,061 parsable synthetic samples	92.75% consistent
HumanEval	Python AST + compile	5,540 code-bearing synthetic samples	99.26% syntax-valid
HumanEval	AST interface matching	5,540 code-bearing synthetic samples	99.08% interface match
HumanEval	Isolated literal-test execution	166 executable synthetic samples	90.96% pass
BFCL	JSON wrapper/schema validation	2,373 synthetic fusion samples	100% format-valid

For the training data synthesized with GSM8K as the downstream target, the rate measures equation-chain consistency among synthetic responses containing extractable explicit equations, rather than formal proof of every open-form solution. Likewise, the execution rate is reported only for synthetic examples generated with HumanEval as the target and containing runnable literal tests.

B.3 Contamination Analysis and Held-Out Generalization

Contamination Analysis. For each final 5k/8k training set synthesized by ANDES for a downstream target task, we compare the synthetic samples against that task's evaluation questions using normalized exact match and candidate retrieval based on 5-gram Jaccard, BM25, ROUGE-L, and BGE-small embedding cosine similarity. The risk thresholds are Jaccard ≥ 0.50, ROUGE-L ≥ 0.80, and cosine ≥ 0.90; the top 50 retrieved pairs for each target task are manually reviewed. We separately measure exact and 5-gram near duplicates within each ANDES-synthesized training set. Table B.3 summarizes the results.

Table B.3: Contamination and internal-duplication analysis across the seven training sets synthesized by ANDES for distinct downstream target tasks, reported for each base-model collection.

Collection	Test exact	Cosine ≥ 0.90	Max exact dup.	Max 5-gram ≥ 0.8
Qwen3-1.7B	0	3 (target: GSM8K; 3/1,319)	0.47%	1.16%
Qwen3-4B	0	0	0.25%	0.71%
SmolLM3-3B	0	0	0.25%	0.71%
Gemma3-4B	0	0	0.25%	0.71%

As shown in Table B.3, no synthetic training set contains an exact match with its corresponding evaluation questions. Manual inspection further confirms that the three cosine-triggered pairs are generic arithmetic schemas with different values, objectives, solution paths, and answers rather than copied questions or answers. The low within-set duplication rates additionally indicate that the gains are unlikely to arise from repeated synthetic templates.

Held-Out Generalization. To test whether the synthesized data improves the underlying mathematical capability rather than only the target evaluation, we fine-tune Qwen3-1.7B on 5k training samples synthesized by ANDES with GSM8K as the downstream target, freeze the resulting checkpoint, and evaluate it on SVAMP and ASDiv-A, neither of which was supplied to the synthesis agent. Comparing the synthetic training set against these held-out evaluation sets yields zero exact matches and zero threshold-flagged pairs. Table B.4 reports the resulting transfer performance.

Table B.4: Transfer to unseen benchmarks not supplied to the synthesis agent.

Held-out set	Base	Condor-style	ANDES	Δ Base	Δ Condor
SVAMP	54.00	83.00	**89.00**	+35.00	+6.00
ASDiv-A	62.30	86.07	**88.52**	+26.22	+2.45

As shown in Table B.4, the ANDES-trained checkpoint reaches 89.00 on SVAMP and 88.52 on ASDiv-A, improving over both the base model and Condor-style training. Together with the zero-overlap analysis, these gains provide evidence that the synthesized data transfers to unseen tasks rather than exploiting target-specific test items.

Table C.2: Experiments results on PostTrainBench (accuracy in %) based on the Qwen3-1.7B base model. The best and second-best results among the displayed agent methods are highlighted in **bold** and underlined. The Official Instruct Model and Base Model rows are references and are excluded from this ranking. Numbers after arrows denote absolute changes over the corresponding zero-shot base model, with improvements in red and declines in green.

Method	AIME 2025	ArenaHard	BFCL	GPQA Main	GSM8K	HealthBench	HumanEval	AVG (%)
	(I) Evaluation with Official Instruct Model							
Official Instruct Model	26.67	50.00	94.00	35.49	88.48	44.92	68.90	49.41
	(II) Evaluation with Base Model (Zero-Shot)							
Base Model (Qwen3-1.7B)	0.00	0.91	0.00	14.06	12.66	7.54	7.93	6.66
	(III) Evaluation with Agent-Post-Trained Base Model							
GPT-5.4 (High)	0.00 (↑0.00)	0.51 (↓0.40)	0.00 (↑0.00)	**29.39** (↑15.33)	50.42 (↑37.76)	15.42 (↑7.88)	30.28 (↑22.35)	17.42 (↑10.76)
GPT-5.2	2.22 (↑2.22)	1.27 (↑0.36)	29.33 (↑29.33)	17.41 (↑3.35)	51.00 (↑38.34)	9.33 (↑1.79)	32.72 (↑24.79)	16.68 (↑10.02)
Gemini-3.1-Pro	0.00 (↑0.00)	2.27 (↑1.36)	85.67 (↑85.67)	16.15 (↑2.09)	39.58 (↑26.92)	10.93 (↑3.39)	36.18 (↑28.25)	19.78 (↑13.12)
Opus-4.6	0.00 (↑0.00)	1.15 (↑0.24)	28.33 (↑28.33)	22.92 (↑8.86)	27.14 (↑14.48)	8.58 (↑1.04)	22.76 (↑14.83)	13.90 (↑7.24)
Opus-4.6 (1M)	5.56 (↑5.56)	3.21 (↑2.30)	87.33 (↑87.33)	22.99 (↑8.93)	42.99 (↑30.33)	9.73 (↑2.19)	37.20 (↑29.27)	22.99 (↑16.33)
Opus-4.7 (xHigh)	**6.67** (↑6.67)	33.84 (↑32.93)	91.33 (↑91.33)	29.32 (↑15.26)	50.47 (↑37.81)	16.23 (↑8.69)	42.28 (↑34.35)	30.17 (↑23.51)
Opus-4.8 (High)	1.67 (↑1.67)	**45.01** (↑44.10)	47.50 (↑47.50)	27.01 (↑12.95)	61.75 (↑49.09)	**31.20** (↑23.66)	38.72 (↑30.79)	29.69 (↑23.03)
Opus-4.8 (Max)	3.33 (↑3.33)	16.81 (↑15.90)	91.00 (↑91.00)	25.89 (↑11.83)	64.40 (↑51.74)	25.27 (↑17.73)	39.02 (↑31.09)	29.70 (↑23.04)
GLM-5.2 (Max)	1.11 (↑1.11)	13.66 (↑12.75)	**95.33** (↑95.33)	28.42 (↑14.36)	**79.61** (↑66.95)	25.04 (↑17.50)	55.28 (↑47.35)	**32.91** (↑26.25)
	(IV) Evaluation with Proposed Method (Compared with Baseline)							
GLM-4.7 (OpenCode)	0.00* (↑0.00)	0.91* (↑0.00)	0.00* (↑0.00)	14.06† (↑0.00)	12.66* (↑0.00)	7.54* (↑0.00)	12.20 (↑4.27)	7.11 (↑0.45)
GLM-4.7 + Trainer Scaffold	0.00 (↑0.00)	5.70 (↑4.79)	43.00 (↑43.00)	20.13 (↑6.07)	36.88 (↑24.22)	16.74 (↑9.20)	44.10 (↑36.17)	19.45 (↑12.79)
GLM-4.7 & ANDES (Ours)	0.00 (↑0.00)	14.22 (↑13.31)	90.16 (↑90.16)	28.48 (↑14.42)	63.19 (↑50.53)	27.08 (↑19.54)	**57.51** (↑49.58)	31.41 (↑24.75)

* Model not submitted – base model score shown † Evaluation error – base model score shown

C Detailed PostTrainBench Results

C.1 Stability Across Independent Runs

Table C.1 reports the mean and standard deviation over four independent end-to-end runs for each of the four base models, with every run covering all seven downstream target tasks.

Table C.1: Mean and standard deviation over four independent end-to-end runs for each base model.

Base Model	AIME 2025	ArenaHard	BFCL	GPQA Main	GSM8K	HealthBench	HumanEval	AVG (%)
Qwen3-1.7B	0.00±0.00	14.22±1.30	90.16±1.60	28.48±0.80	63.19±1.20	27.08±1.40	57.51±2.00	31.41±0.70
Qwen3-4B	9.00±1.70	51.94±0.85	91.22±0.60	31.40±2.20	83.63±1.10	31.07±1.45	66.62±1.55	41.21±0.66
SmolLM3-3B	6.60±1.20	15.21±1.35	79.91±1.55	27.00±1.10	68.67±1.45	38.50±1.70	42.42±1.85	32.91±0.78
Gemma3-4B	0.00±0.00	26.55±1.40	86.09±1.25	26.83±0.95	77.92±1.50	34.46±1.60	33.59±1.35	32.05±0.72

The consistently small standard deviations, including aggregate-score deviations between 0.66 and 0.78, indicate that the performance gains remain stable across independent runs and base-model settings.

C.2 Results Across Base Models

Tabs. C.2, C.3, C.4, and C.5 report the detailed PostTrainBench results under four base-model settings. Overall, ANDES brings consistent improvements across different model families and different initial capability levels. For Qwen3-1.7B, ANDES improves the average score from 7.11% of the GLM-4.7 baseline to 31.41%, exceeding Opus-4.7 at 30.17% while trailing GLM-5.2 (Max) at 32.91%. For Qwen3-4B, ANDES obtains 41.21%, substantially above the GLM-4.7 baseline of 14.70% and close to Opus-4.8 (High) at 41.34%, while GLM-5.2 (Max) reaches 44.89%. For SmolLM3-3B, ANDES reaches 32.91%, 0.06 points below Opus-4.8 (High) at 32.97%. For Gemma3-4B, ANDES reaches 32.05%, below Opus-4.8 (Max) at 34.73% and above Opus-4.8 (High) at 31.17%.

Table C.3: Experiments results on PostTrainBench (accuracy in %) based on the Qwen3-4B base model. The best and second-best results among the displayed agent methods are highlighted in **bold** and underlined. The Official Instruct Model and Base Model rows are references and are excluded from this ranking. Numbers after arrows denote absolute changes over the corresponding zero-shot base model, with improvements in red and declines in green.

Method	AIME 2025	ArenaHard	BFCL	GPQA Main	GSM8K	HealthBench	HumanEval	AVG (%)
(I) Evaluation with Official Instruct Model								
Official Instruct Model	53.33	86.84	95.00	44.64	93.78	52.72	77.44	63.75
(II) Evaluation with Base Model (Zero-Shot)								
Base Model (Qwen3-4B)	3.33	3.42	0.00	13.39	41.85	13.38	36.59	14.34
(III) Evaluation with Agent-Post-Trained Base Model								
GPT-5.4 (High)	2.22 (↓1.11)	9.71 (↑6.29)	31.00 (↑31.00)	23.96 (↑10.57)	51.58 (↑9.73)	18.17 (↑4.79)	25.81 (↓10.78)	19.99 (↑5.65)
GPT-5.2	1.11 (↓2.22)	5.47 (↑2.05)	58.33 (↑58.33)	23.07 (↑9.68)	64.59 (↑22.74)	11.79 (↓1.59)	37.40 (↑0.81)	22.47 (↑8.13)
Gemini-3.1-Pro	2.22 (↓1.11)	4.98 (↑1.56)	57.33 (↑57.33)	20.16 (↑6.77)	47.54 (↑5.69)	8.82 (↓4.56)	52.03 (↑15.44)	21.35 (↑7.01)
Opus-4.6	5.56 (↑2.23)	6.34 (↑2.92)	96.67 (↑96.67)	25.30 (↑11.91)	52.19 (↑10.34)	22.87 (↑9.49)	36.59 (↑0.00)	27.71 (↑13.37)
Opus-4.6 (1M)	6.67 (↑3.34)	5.55 (↑2.13)	**97.33** (↑97.33)	29.91 (↑16.52)	78.75 (↑36.90)	10.71 (↓2.67)	57.93 (↑21.34)	31.48 (↑17.14)
Opus-4.7 (xHigh)	7.78 (↑4.45)	21.90 (↑18.48)	62.00 (↑62.00)	24.03 (↑10.64)	63.76 (↑21.91)	21.72 (↑8.34)	62.60 (↑26.01)	30.38 (↑16.04)
Opus-4.8 (High)	**23.33** (↑20.00)	41.56 (↑38.14)	48.50 (↑48.50)	24.44 (↑11.05)	85.06 (↑43.21)	41.61 (↑28.23)	71.34 (↑34.75)	41.34 (↑27.00)
Opus-4.8 (Max)	**23.33** (↑20.00)	22.64 (↑19.22)	96.00 (↑96.00)	23.21 (↑9.82)	**89.88** (↑48.03)	**41.84** (↑28.46)	69.21 (↑32.62)	43.17 (↑28.83)
GLM-5.2 (Max)	16.67 (↑13.34)	**54.25** (↑50.83)	97.00 (↑97.00)	**32.89** (↑19.50)	85.29 (↑43.44)	30.04 (↑16.66)	**76.02** (↑39.43)	**44.89** (↑30.55)
(IV) Evaluation with Proposed Method (Compared with Baseline)								
GLM-4.7 (OpenCode)	3.33* (↑0.00)	3.42* (↑0.00)	0.00* (↑0.00)	13.39* (↑0.00)	45.72 (↑3.87)	13.38* (↑0.00)	36.59* (↑0.00)	14.70 (↑0.36)
GLM-4.7 + Trainer Scaffold	0.00 (↓3.33)	34.41 (↑30.99)	63.00 (↑63.00)	22.10 (↑8.71)	67.55 (↑25.70)	9.71 (↓3.67)	56.70 (↑20.11)	26.90 (↑12.56)
GLM-4.7 & ANDES (Ours)	9.00 (↑5.67)	51.94 (↑48.52)	91.22 (↑91.22)	31.40 (↑18.01)	83.63 (↑41.78)	31.07 (↑17.69)	66.62 (↑30.03)	41.21 (↑26.87)

* Model not submitted – base model score shown † Evaluation error – base model score shown

These results show that the effectiveness of ANDES does not depend on a specific base model. Across Qwen, SmolLM, and Gemma model families, ANDES can synthesize useful training data according to the target benchmark requirements. Moreover, the improvements cover diverse tasks, including mathematical reasoning, tool use, code generation, health dialogue, and open-ended writing. This indicates that ANDES does not merely optimize for a single benchmark, but provides general and transferable training signals across different tasks. Across all four base models, ANDES improves the weighted average over GLM-4.7 using the same trainer scaffold without ANDES, and it performs better on most individual model–task pairs. This further substantiates the net gains that ANDES contributes to the trainer agent.

Table C.4: Experiments results on PostTrainBench (accuracy in %) based on the SmolLM3-3B base model. The best and second-best results among the displayed agent methods are highlighted in **bold** and underlined. The Official Instruct Model and Base Model rows are references and are excluded from this ranking. Numbers after arrows denote absolute changes over the corresponding zero-shot base model, with improvements in red and declines in green.

Method	AIME 2025	ArenaHard	BFCL	GPQA Main	GSM8K	HealthBench	HumanEval	AVG (%)
(I) Evaluation with Official Instruct Model								
Official Instruct Model	26.67	49.20	84.00	33.26	82.18	29.58	70.12	44.81
(II) Evaluation with Base Model (Zero-Shot)								
Base Model (SmolLM3-3B)	3.33	0.42	0.00	4.91	21.08	0.00	6.10	4.52
(III) Evaluation with Agent-Post-Trained Base Model								
GPT-5.4 (High)	0.00 (↓3.33)	14.34 (↑13.92)	29.67 (↑29.67)	29.02 (↑24.11)	50.14 (↑29.06)	18.64 (↑18.64)	24.19 (↑18.09)	20.72 (↑16.20)
GPT-5.2	0.00 (↓3.33)	6.49 (↑6.07)	33.33 (↑33.33)	25.82 (↑20.91)	56.08 (↑35.00)	21.92 (↑21.92)	29.27 (↑23.17)	21.26 (↑16.74)
Gemini-3.1-Pro	12.22 (↑8.89)	6.42 (↑6.00)	27.67 (↑27.67)	18.01 (↑13.10)	55.42 (↑34.34)	18.45 (↑18.45)	38.01 (↑31.91)	22.08 (↑17.56)
Opus-4.6	14.44 (↑11.11)	9.00 (↑8.58)	86.67 (↑86.67)	26.41 (↑21.50)	58.05 (↑36.97)	21.12 (↑21.12)	25.61 (↑19.51)	28.52 (↑24.00)
Opus-4.6 (1M)	1.11 (↓2.22)	4.22 (↑3.80)	59.67 (↑59.67)	28.12 (↑23.21)	46.90 (↑25.82)	19.26 (↑19.26)	17.48 (↑11.38)	21.19 (↑16.67)
Opus-4.7 (xHigh)	10.00 (↑6.67)	10.21 (↑9.79)	62.33 (↑62.33)	27.83 (↑22.92)	66.11 (↑45.03)	6.69 (↑6.69)	35.57 (↑29.47)	25.28 (↑20.76)
Opus-4.8 (High)	11.67 (↑8.34)	20.89 (↑20.47)	**97.00** (↑97.00)	4.91 (↑0.00)	**76.69** (↑55.61)	38.39 (↑38.39)	**55.18** (↑49.08)	**32.97** (↑28.45)
Opus-4.8 (Max)	**16.67** (↑13.34)	**37.19** (↑36.77)	50.00 (↑50.00)	17.63 (↑12.72)	51.52 (↑30.44)	32.63 (↑32.63)	28.96 (↑22.86)	28.73 (↑24.21)
GLM-5.2 (Max)	13.33 (↑10.00)	22.71 (↑22.29)	64.00 (↑64.00)	**29.76** (↑24.85)	64.06 (↑42.98)	29.04 (↑29.04)	36.18 (↑30.08)	31.71 (↑27.19)
(IV) Evaluation with Proposed Method (Compared with Baseline)								
GLM-4.7 (OpenCode)	3.33* (↑0.00)	0.42* (↑0.00)	0.00† (↑0.00)	4.91* (↑0.00)	10.54 (↓10.54)	0.00* (↑0.00)	6.10* (↑0.00)	3.53 (↓0.99)
GLM-4.7 + Trainer Scaffold	0.00 (↓3.33)	17.32 (↑16.90)	70.00 (↑70.00)	19.31 (↑14.40)	48.36 (↑27.28)	7.38 (↑7.38)	29.27 (↑23.17)	20.12 (↑15.60)
GLM-4.7 & ANDES (Ours)	6.60 (↑3.27)	15.21 (↑14.79)	79.91 (↑79.91)	27.00 (↑22.09)	68.67 (↑47.59)	**38.50** (↑38.50)	42.42 (↑36.32)	32.91 (↑28.39)

* Model not submitted – base model score shown † Evaluation error – base model score shown

Table C.5: Experiments results on PostTrainBench (accuracy in %) based on the Gemma3-4B base model. The best and second-best results among the displayed agent methods are highlighted in **bold** and underlined. The Official Instruct Model and Base Model rows are references and are excluded from this ranking. Numbers after arrows denote absolute changes over the corresponding zero-shot base model, with improvements in red and declines in green.

Method	AIME 2025	ArenaHard	BFCL	GPQA Main	GSM8K	HealthBench	HumanEval	AVG (%)
(I) Evaluation with Official Instruct Model								
Official Instruct Model	10.00	94.80	67.00	31.47	83.55	46.06	69.51	46.58
(II) Evaluation with Base Model (Zero-Shot)								
Base Model (Gemma3-4B)	0.00	0.29	6.00	1.56	6.14	17.04	0.61	4.60
(III) Evaluation with Agent-Post-Trained Base Model								
GPT-5.4 (High)	0.00 (↑0.00)	15.71 (↑15.42)	63.67 (↑57.67)	**29.54** (↑27.98)	40.59 (↑34.45)	16.92 (↓0.12)	29.07 (↑28.46)	22.81 (↑18.21)
GPT-5.2	0.00 (↑0.00)	13.20 (↑12.91)	89.00 (↑83.00)	28.57 (↑27.01)	51.91 (↑45.77)	20.19 (↑3.15)	21.54 (↑20.93)	25.11 (↑20.51)
Gemini-3.1-Pro	**1.11** (↑1.11)	16.01 (↑15.72)	80.67 (↑74.67)	19.79 (↑18.23)	39.50 (↑33.36)	19.70 (↑2.66)	34.55 (↑33.94)	23.15 (↑18.55)
Opus-4.6	0.00 (↑0.00)	14.61 (↑14.32)	92.00 (↑86.00)	27.46 (↑25.90)	26.79 (↑20.65)	22.66 (↑5.62)	14.02 (↑13.41)	22.52 (↑17.92)
Opus-4.6 (1M)	0.00 (↑0.00)	13.95 (↑13.66)	64.33 (↑58.33)	28.12 (↑26.56)	36.42 (↑30.28)	21.49 (↑4.45)	36.38 (↑35.77)	23.60 (↑19.00)
Opus-4.7 (xHigh)	**1.11** (↑1.11)	30.86 (↑30.57)	91.33 (↑85.33)	28.72 (↑27.16)	55.70 (↑49.56)	21.49 (↑4.45)	27.85 (↑27.24)	28.43 (↑23.83)
Opus-4.8 (High)	0.00 (↑0.00)	18.93 (↑18.64)	**97.00** (↑91.00)	28.68 (↑27.12)	55.27 (↑49.13)	29.85 (↑12.81)	**48.17** (↑47.56)	31.17 (↑26.57)
Opus-4.8 (Max)	0.00 (↑0.00)	**47.39** (↑47.10)	**97.00** (↑91.00)	28.35 (↑26.79)	69.67 (↑63.53)	**41.98** (↑24.94)	24.39 (↑23.78)	**34.73** (↑30.13)
GLM-5.2 (Max)	0.00 (↑0.00)	26.65 (↑26.36)	67.00 (↑61.00)	24.63 (↑23.07)	61.87 (↑55.73)	29.52 (↑12.48)	32.72 (↑32.11)	27.64 (↑23.04)
(IV) Evaluation with Proposed Method (Compared with Baseline)								
GLM-4.7 (OpenCode)	0.00† (↑0.00)	0.29* (↑0.00)	6.00† (↑0.00)	1.56† (↑0.00)	6.14† (↑0.00)	17.04* (↑0.00)	0.61† (↑0.00)	4.60 (↑0.00)
GLM-4.7 + Trainer Scaffold	0.00 (↑0.00)	31.12 (↑30.83)	61.00 (↑55.00)	22.76 (↑21.20)	36.16 (↑30.02)	4.89 (↓12.15)	28.29 (↑27.68)	19.76 (↑15.16)
GLM-4.7 & ANDES (Ours)	0.00 (↑0.00)	26.55 (↑26.26)	86.09 (↑80.09)	26.83 (↑25.27)	**77.92** (↑71.78)	34.46 (↑17.42)	33.59 (↑32.98)	32.05 (↑27.45)

* Model not submitted – base model score shown † Evaluation error – base model score shown

D WORLD TREE COVERAGE

Analysis on World Tree Coverage. Fig. A.1 first shows the knowledge distribution of all nodes in the world tree. The initialized world tree spans broad macro categories, including healthcare and life sciences, mathematics and natural sciences, computing and information security, engineering and technology, education and personal development, business and governance, humanities and arts, and lifestyle services. Each category contains rich topic, theme, and scenario nodes, forming a broad and hierarchical synthesis space. Therefore, ANDES does not rely on a single domain or fixed templates for data generation. Instead, it can perform target-aware routing and dynamic expansion over a wide knowledge space, which supports the diversity, generality, and cross-benchmark adaptability of the synthesized data.

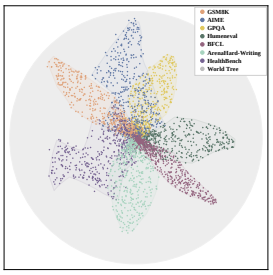


Figure D.1: **t-sne visualization of the question of the ANDES world tree.** It shows that the world tree provides a broad and general knowledge space that supports diverse target tasks.

Fig. D.1 further visualizes the question embeddings of synthetic data generated for different PostTrainBench benchmarks. The samples from different benchmarks form distinguishable regions in the embedding space while still sharing overlapping areas near the center. This indicates that the world tree of ANDES covers diverse capability directions, including mathematical reasoning, code generation, tool use, health dialogue, and open-ended writing. Meanwhile, the benchmark-specific distributions show that the directed routing mechanism can select suitable topics, themes, and scenarios according to the target task, thereby generating data aligned with the required capabilities.

E CASE STUDY: AGENT TRAJECTORY

In this section, we present a representative agent trajectory of the trainer agent utilizing the ANDES skill across multiple benchmarks. As illustrated below, the agent does not merely overfit to the surface-level patterns or specific formatting of individual benchmarks. Instead, guided by the ANDES skill, the agent autonomously reconstructs the target requirements into a set of refined macro-cognitive domains.

Specifically, the agent decomposes complex benchmark prompts into fundamental operational capabilities, such as “Quantitative Modeling and Relation Extraction” for mathematics or “Edge Cases and Robust Programming” for code synthesis. By reorganizing task-local prompts into these macro domains, ANDES ensures that the synthesized data captures the underlying reasoning logic rather than memorizing task-specific shortcuts. This hierarchical abstraction allows the agent to maintain high fidelity to the target benchmarks while simultaneously fostering robust cross-task generalization, effectively preventing diversity collapse during long-horizon autonomous alignment.

E1. Initial Input and To-Do List

This module defines the run contract before any synthesis begins: the target benchmark, the base model, the hard constraints, and the execution checklist. It anchors the rest of the trajectory so the reader can interpret later decisions as part of one controlled GSM8K post-training run.

E1. Initial Input and To-Do List

Initial input.
(a) Target benchmark: GSM8K;
(b) Base model: Qwen3-4B-Base;
(c) Objective: Fine-tune base model for grade-school arithmetic word-problem reasoning;

Hard constraints.
(a) Finish within 10 hours;
(b) Fine-tune the base model only;
(c) Avoid GSM8K test leakage;
(d) Synthesize data through ANDES only;
(e) Retain ∼8,000 Alpaca-format samples;

Execution to-do list.
(a) Analyze GSM8K requirements and divide them into cognitive-operation domains;
(b) Use ANDES to synthesize domain-targeted training data;
(c) Merge retained samples into Alpaca instruction/input/output format;
(d) Run full SFT with LlamaFactory on Qwen3-4B-Base;
(e) Evaluate the best checkpoint and save it as final model;

E2. Capability-Domain Agent Trajectory

This module merges the macro capability-domain split with the full interaction trajectory. The domain split jointly considers the model abilities that need strengthening, the reasoning demands expressed by the benchmark content, and the post-training coverage needed for robust generalization. The agent then turns these domains into synthesis specifications, delegates generation to ANDES, reads reports as feedback signals, and uses each report to control retention and sharpen the next task description. This report-driven loop connects data synthesis, quality control, aggregation, training, and evaluation into a complete automated pipeline for high-quality post-training.

E2. Capability-Domain Agent Trajectory

Boot the agent shell. (a) The trainer agent loads the post-training, data-synthesis, and training-configuration skills; (b) Parses the benchmark instruction and resource constraints from the task prompt; (c) Establishes a sandboxed execution plan with synthesis, training, and evaluation phases.

Research the benchmark. (a) The agent inspects GSM8K-style reasoning requirements rather than copying benchmark examples; (b) Identifies the core capability demand: arithmetic word-problem solving with chained intermediate states; (c) Converts surface problem formats into abstract cognitive operations such as extraction, modeling, constraint checking, and answer verification.

Construct the domain plan. (a) The agent partitions the benchmark into non-overlapping macro capability domains; (b) Each domain is defined by reasoning structure rather than topic keywords; (c) The plan is checked for coverage balance so that no single arithmetic pattern dominates later synthesis.

Author the synthesis specification. (a) For each domain, the agent writes a task description τ_j that states the target capability, difficulty style, and constraint structure; (b) It assigns a synthesis budget N_j and a fixed format protocol ϕ_j; (c) It records expected failure modes, including overly narrow descriptions, template lock-in, constraint drift, and shallow form-only reasoning.

Delegate synthesis to ANDES. (a) The agent invokes ANDES as an external synthesis tool rather than as a planner; (b) ANDES routes each domain through the world-tree hierarchy from topic to theme to scenario; (c) Fusion-track samples inject the domain requirement into diverse scenarios, while generic-track samples preserve broad contextual coverage.

Read reports and decide retention. (a) After each domain run, the agent reads the synthesis report together with the original task description; (b) It compares topic coverage, effective quantity, fusion/generic balance, logical diversity, and reported signals; (c) It chooses a predefined random-discard tier only after this report–description comparison, avoiding any manual semantic filtering. *The following Round1–Round3 blocks are fully expanded examples of this interaction paradigm. Each round shows how one task description is instantiated into a synthesized case, how the* ANDES *report exposes quality signals, and how the agent converts that report into a random-discard decision. The rounds are interactive rather than independent: the report from one round informs how the agent sharpens the next round's task description, coverage emphasis, and expected failure modes. The final "Later Rounds" block summarizes the remaining task-description/domain pairs, with ellipses marking omitted case-level examples.*

Round1

Task description. Generate GSM8K-style arithmetic tasks that require a solver to maintain a running state across several dependent commercial operations, including discounts, giveaway rules, production costs, and final profit computation. The task should reward ordered reasoning rather than isolated arithmetic.
Domain1: Multi-step Sequential Reasoning: Generate arithmetic word problems that require 3 to 7 ordered operations. Each later operation depends on an earlier intermediate result, so the solver must maintain a running state rather than execute isolated arithmetic.
Full task. A startup sells eco-friendly bottles. Each bottle costs $8 to produce and normally sells for $20. During a promotion, the first 100 sold bottles receive a 25% discount. In addition, for every 5 bottles sold during the promotion, 1 extra bottle is given away for free. After the promotion, another 150 bottles are sold at the regular price. Compute total revenue and total profit.
Capability focus. The case requires a chained calculation: discounted price, promotional revenue, free-bottle count, production cost, regular-sale revenue, and final profit.
Reasoning. The discounted price is $20 × (1 - 0.25) = $15. Promotion revenue is 100 × $15 = $1,500. The free-bottle rule gives 100 / 5 = 20 free bottles, so the promotion requires producing 100 + 20 = 120 bottles, costing 120 × $8 = $960. After the promotion, regular revenue is 150 × $20 = $3,000, and regular production cost is 150 × $8 = $1,200. Total revenue is $1,500 + $3,000 = $4,500, total cost is $960 + $1,200 = $2,160, and profit is $4,500 - $2,160 = $2,340.
Answer. Total revenue is $4,500, and total profit is $2,340.
Report. The synthesis report shows strong coverage of sequential arithmetic in commerce, promotion, and inventory scenarios. Most samples preserve a clear dependency chain from intermediate quantities to final profit, and the fusion track adds proportional details such as discounts and bonus units without overwhelming the D1 target. No severe coverage collapse is detected, but a small cluster of examples repeats the revenue–cost–profit template with similar ordering.
Agent report analysis and retention decision. The agent compares the report with the original task description and finds the intended multi-step dependency structure intact. Because the only material risk is mild template repetition rather than semantic drift, the agent chooses a 10% random discard ratio and carries the anti-template warning into the next round's task description without performing content-based filtering.

⇓

Round2

Task description. Generate GSM8K-style tasks where the solver must extract the relevant quantities from natural language, ignore distractor objects, align comparable categories, and aggregate matched, overpacked, or underpacked counts. The task should test modeling from text before calculation.
Domain2: Information Extraction and Mathematical Modeling: Extract multiple quantities and relationships from natural language, ignore distractors, and translate the remaining evidence into equations. The same surface pattern may map to different mathematical structures depending on context.
Full task. A travel guide recommends packing 10 shirts, 5 pairs of pants, 2 jackets, 8 pairs of socks, and 3 pairs of shoes. The traveler also packs 2 hats, 1 scarf, 4 books, and 1 camera. The actual relevant items are 12 shirts, 6 pairs of pants, 1 jacket, 10 pairs of socks, and 2 pairs of shoes. Compute correctly packed, overpacked, and underpacked items.
Capability focus. The case tests whether the model can ignore distractors and compare only the categories specified by the guide.
Reasoning. Shirts: 10 recommended and 12 packed, so 10 correct and 2 overpacked. Pants: 5 recommended and 6 packed, so 5 correct and 1 overpacked. Jackets: 2 recommended and 1 packed, so 1 correct and 1 underpacked. Socks: 8 recommended and 10 packed, so 8 correct and 2 overpacked. Shoes: 3 recommended and 2 packed, so 2 correct and 1 underpacked. Thus, correctly packed items are 10 + 5 + 1 + 8 + 2 = 26, overpacked items are 2 + 1 + 0 + 2 + 0 = 5, and underpacked items are 0 + 0 + 1 + 0 + 1 = 2. The extra items are 2 + 1 + 4 + 1 = 8 distractors and do not affect the requested totals.
Answer. Correctly packed: 26; overpacked: 5; underpacked: 2.
Report. The synthesis report indicates that D2 samples consistently require extraction before computation. Generated cases vary across travel, classroom, shopping, and household settings, while distractor quantities are usually separable from the categories requested by the task. The report finds no major template lock-in and no constraint drift; most samples keep the answer grounded in an explicit category comparison rather than an unstructured count of all mentioned objects.
Agent report analysis and retention decision. The agent reads the report against the task description and confirms that distractor suppression and mathematical modeling are the dominant operations. Since failure signals are low and the effective quantity is near the planned budget, the agent chooses a 0% random discard ratio and uses the clean extraction signal to make the next task description emphasize explicit constraint verification.

⇓

Round3

Task description. Generate GSM8K-style tasks requiring a feasibility or threshold judgment after arithmetic computation. The solver must compute intermediate quantities, compare them against an explicit constraint, and revise or verify the result when the original condition fails.
Domain3: Constraint Satisfaction and Logical Reasoning: Reason under explicit or implicit constraints: maximum or minimum values, budget limits, time limits, capacities, integer requirements, and interacting conditions. The answer must be feasible, not merely numerically computed.
Full task. A bookstore membership program gives members a 15% discount. Members spend \$50 per visit on average. The non-discounted profit margin is 30%. Compute the member profit per visit and verify whether the discounted margin remains at least 20%. If it fails, suggest a revised discount.
Capability focus. The case requires proportional reasoning over revenue and cost, then a constraint check against a margin threshold and a verification of the revised discount.
Reasoning. The non-discounted cost is \$50 × (1 - 0.30) = \$35. With a 15% discount, member revenue is \$50 × (1 - 0.15) = \$42.50, so profit is \$42.50 - \$35 = \$7.50. The resulting margin is \$7.50 / \$42.50 ≈ 17.65%, which is below 20%. Let the revised discount be x. To meet the 20% margin threshold, $(50(1-x)-35)/(50(1-x)) = 0.20$, which gives $50 - 50x - 35 = 10 - 10x$, then $5 = 40x$, so $x = 0.125$. A 12.5% discount gives revenue \$50 × 0.875 = \$43.75, profit \$43.75 - \$35 = \$8.75, and margin \$8.75 / \$43.75 = 20%.
Answer. Under the 15% discount, profit is \$7.50 and margin is about 17.65%, so the constraint fails. A 12.5% discount reaches the 20% threshold; for strictly above 20%, the discount should be slightly below 12.5%.
Report. The synthesis report shows good coverage of budget, margin, capacity, and threshold checks, with many samples requiring the model to reject a numerically computed value when it violates a stated condition. However, the report also flags a moderate F5-style constraint-drift risk: some generated cases compute a revised value but do not explicitly verify that the revised value satisfies the original threshold. Logical diversity is acceptable, but the verification step is less consistently represented than the arithmetic step.
Agent report analysis and retention decision. The agent compares these findings with the D3 task description and treats verification omission as directly relevant to the target capability. Because the batch still contains useful constraint-reasoning data but has a moderate alignment risk, the agent chooses a 20% random discard ratio while preserving the remaining records verbatim.

⇓

Later Rounds

Task description. Generate GSM8K-style tasks that require ratios, rates, scaling, unit conversion, percentages, and quantity comparisons. The task should force the solver to preserve proportional relationships while moving across units or scales.
Domain4: Proportional and Comparative Reasoning: Handle ratios, rates, scaling, unit conversion, percentages, and comparisons between quantities. The key skill is preserving the proportional relationship while quantities move across units or scales.

.....

Task description. Generate GSM8K-style tasks that require decomposing a multi-component situation into ordered sub-problems, solving each piece, and recombining the partial results into one final answer.
Domain5: Complex Problem Decomposition: Break a multi-component task into 2 to 4 sub-problems, order the dependent pieces, solve them separately, and recombine partial answers. The domain targets planning structure as much as computation.

.....

Task description. Generate GSM8K-style tasks that require checking intermediate and final answers against the original constraints, detecting invalid outputs, and correcting the result when practical sanity conditions fail.
Domain6: Verification and Self-Correction: Check intermediate and final answers against the original constraints and practical sanity conditions. Examples include whole-number checks, capacity checks, budget checks, and detecting impossible configurations.

.....

Complete the synthesis loop. (a) The agent repeats the domain-level call, report reading, and discard-decision loop until all initial domains are completed; (b) If the retained pool is still insufficient, it adds supplementary capability domains instead of resampling the same domain; (c) Throughout the loop, the format protocol remains fixed and retained samples are not rewritten, deduplicated, or filtered by content.

Aggregate training data. (a) The agent merges all retained domain outputs into one Alpaca-style dataset; (b) Both fusion and generic records are preserved as valid training signals; (c) The final dataset reflects the original capability split while maintaining scenario diversity from the world tree.

Train and evaluate. (a) The agent launches full supervised fine-tuning on the selected base model; (b) Training progress is monitored through logs and checkpoint artifacts; (c) The final checkpoint is evaluated on the target benchmark to measure whether the synthesized curriculum improves reasoning performance.