I-EDI: Robust Self-Evolution Agents via Verifiable Counterfactual Simulation

Runze Fan[1] , **Yong Li**[1*]
[1]College of Computer Science, Chongqing University
fanrz@stu.cqu.edu.cn, yongli@cqu.edu.cn

Abstract

Current self-evolution methods mostly chase recall, they learn from any trajectory that ends with the right answer, even when the path relies on a "lucky guess". Such reasoning appears valid in-distribution but often fails the moment the task shifts slightly. We argue that robust evolution implies Structural Invariance: a reasoning path is valid only if its core dependency graph remains isomorphic under counterfactual perturbations. I-EDI enforces this with Verifiable Counterfactual Simulation (VCS). Instead of treating data generation as mere "sample and filter," we treat it as an intervention. Unlike standard rejection sampling, VCS acts as a structural stress test: it accepts a reasoning trace only if it remains valid across generated counterfactuals (e.g., numerical perturbations, context shifts). While this strict filtering reduces the volume of training data (trading recall for precision), our experiments on MATH and GSM8K show it effectively prevents hallucination accumulation, achieving superior out-of-distribution generalization compared to baselines.

1 Introduction

LLM agents[Team *et al.*, 2023; Yang *et al.*, 2025a] have gotten better at logic[Wu *et al.*, 2025a] and tool use[Luo *et al.*, 2025], largely because of Chain-of-Thought (CoT)[Wei *et al.*, 2022] and self-correction. But the current approach to self-evolution has a blind spot: Most self-evolution methods still put too much faith in the final answer. Existing frameworks lean almost entirely on outcome supervision.[Rein *et al.*, 2024; Lu *et al.*, 2023; Chen, 2021]. If an agent lands on the correct answer, the system treats the reasoning path as valid—even if the logic was flawed or simply a "lucky guess"[Sumers *et al.*, 2024]. Since standard rejection sampling only checks the last output, these structurally unsound trajectories slip into the training set. This isn't harmless noise, it's poison for the model. Instead of learning robust decision-making, the agent effectively memorizes bad heuristics, creating a brittle policy that falls apart the moment the data distribution shifts[An *et al.*, 2024]. Recent work has pushed toward modular agent designs, trying to impose structure on reasoning[Wu *et al.*, 2025b; Yang *et al.*, 2025b], but structure isn't enough—verifying the internal logic of the generated data remains the bottleneck. However, defining "correct reasoning" in open-ended domains is messy and subjective. That is why we explicitly focus on Verifiable Domains (like Math and Code) where ground truth is unambiguous and failure is easy to detect. before agents take on fuzzy, ambiguous problems, they should prove they can maintain causal consistency in deterministic ones. They need to show they can isolate which steps are genuinely necessary and which are incidental. In other words, the logic should survive small perturbations to the input, not collapse because the agent memorized a single path that happened to work once.

We present Iterative Event-Driven Internalization (I-EDI), a closed-loop framework designed for enforcing structural consistency in self-evolution. Our main idea is that true understanding requires Structural Invariance. Verifiable Counterfactual Simulation (VCS) is our core mechanism. Instead of broadly accepting all correct trajectories, VCS subjects the agent's reasoning to a "stress test": we generate counterfactual variants of a problem (e.g., numerical perturbations, isomorphic context shifts) and require the agent to solve them with the same core action dependency graph. This acts as a strict filter: if an action sequence is truly causal, it must remain valid across such structural variations.

Figure 1 gives a general picture of the cognitive changes brought by I-EDI, which shows how a novice agent that depends on wordy deliberation transforms into an expert agent with a skillful and precise tool-use mode after it accumulates experience. Unlike static distillation pipelines, I-EDI runs as a spiral loop rather than a one-shot process. Each cycle revisits and refines the agent's behavior through three recurring phases: (1) **Active Boundary Probing**: the agent creates adversarial tasks specifically designed to challenge its current intuitive policy, aiming at the "zone of proximal development." (2) **Verifiable Counterfactual Simulation (VCS)**: to reduce hallucination, we use a deterministic verifier $\mathcal{V}$ over auditable outcomes and admit only trajectories that are strictly verified and structure-consistent. (3) **Parsimonious Internalization**: rather than replicating long thinking processes, agents distill verified trajectories into optimized actions, learning to bypass redundant computations

*Corresponding author.

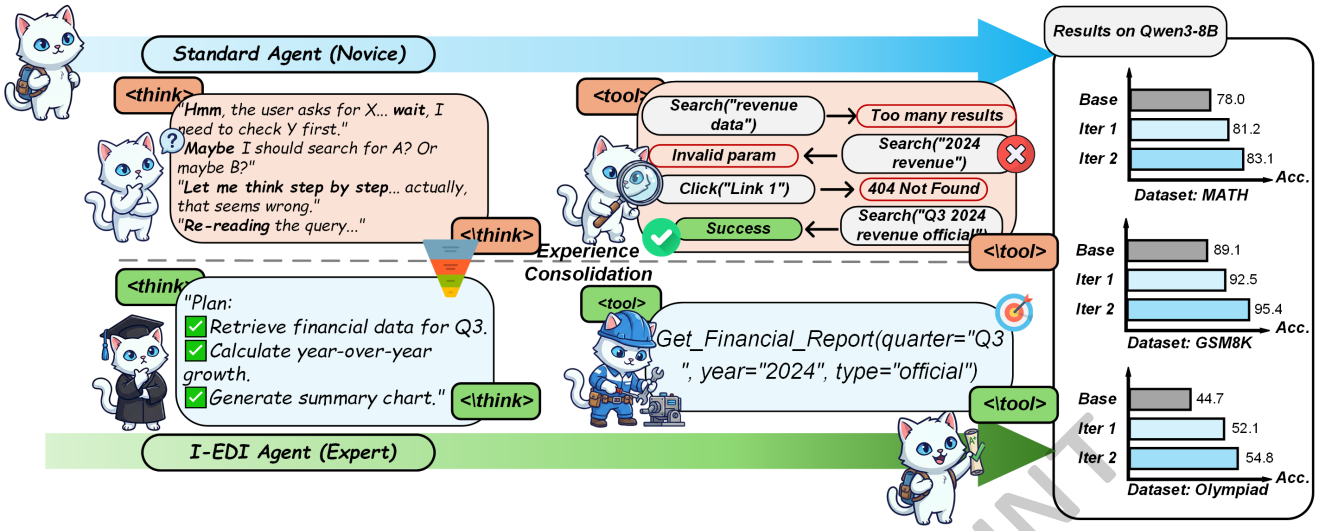

Figure 1: From Novice Deliberation to Expert Intuition via Iterative Event-Driven Internalization. It contrasts a baseline agent stuck in verbose System-2 reasoning and trial-and-error tool use with an I-EDI agent that distills verified experience into a lean, parametric intuition.

and solve problems with the lowest cognitive load. Experiments show that I-EDI consistently outperforms strong self-evolution baselines, with the largest gains on frontier/hard subsets and improved out-of-distribution generalization. Importantly, I-EDI is intentionally built from three auditable primitives:(i) **frontier targeting** (find dissonance where π_{fast} fails but π_{slow} verifies), (ii) **structural stress-test verification** (VCS: Execution + TSC + NbD), and (iii) **parsimonious projection** from τ_{slow} to τ_{opt}. All other components (e.g., edit operators, dedup rules) are **implementation-level choices** that only instantiate these primitives; they are not additional conceptual modules. This reframing makes the method easy to audit: each primitive has a single, executable acceptance criterion.

2 Related Work

2.1 Self-evolving Agent

Research has shifted from static models to agents that can adapt over time[Gao *et al.*, 2025]. One stream formulates prompt engineering as black-box optimization[Zhou *et al.*, 2022; Wang *et al.*, 2023b; Pryzant *et al.*, 2023], iteratively updating instructions to improve performance, while leaving the agent's parameters unchanged. Other approaches focus on accumulating experience, building structured skill libraries or curated toolsets for reuse[Wang *et al.*, 2023a; Wu *et al.*, 2024; Qian *et al.*, 2024; Qiu *et al.*, 2025]. Self-reflexive frameworks instead rely on feedback signals to iteratively correct errors.[Shinn *et al.*, 2023; Liang *et al.*, 2025]. While Agent0[Xia *et al.*, 2025]/R-Zero[Huang *et al.*, 2025]/Socratic-Zero[Wang *et al.*, 2025a] primarily filter by final-answer correctness, I-EDI changes the object of verification: we verify auditable structure via (i) invariance across counterfactual stress tests (TSC) and (ii) operational necessity (NbD under deterministic replay). This targets a concrete failure mode—answer-correct but structurally spurious "lucky guesses"—that accumulates in outcome-only self-training. I-EDI also rethinks what distillation should copy. Instead of imitating verbose rationales, accepted traces are projected to τ_{opt}, which makes training noticeably more stable across rounds.

2.2 Reasoning-Oriented, Distillation, and Efficiency-Driven Works

Much of the prior work improves LLM reasoning by adding CoT or structured planning at inference time, often by generating long multi-step traces. Knowledge distillation can transfer performance, and recent rationale-distillation methods attempt to inherit reasoning behaviors [Do *et al.*, 2025; Wang *et al.*, 2025b; Lee *et al.*, 2024], but they largely perform inference-time scaffolding or imitate surface traces rather than the underlying decision logic. By comparison, hallucination mitigation is usually handled through post-hoc filtering or verification [Lin *et al.*, 2025], not by folding verified experience back into the model.

3 Preliminaries

We consider an agent that solves reasoning tasks by interacting with an environment. Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ the answer space. A policy $\pi_\theta(a_t \mid s_t)$ produces a trajectory $\tau = (x, a_{1:T}, o_{1:T}, y)$, where a_t denotes an action (reasoning step or tool call) and o_t is the resulting observation. We use the same model parameters θ but distinguish two inference modes by decoding: an intuitive policy π_{fast} (greedy decoding) and a deliberate policy π_{slow} that applies a deliberative decoding strategy $\mathbb{D}_{\text{slow}}$ (e.g., exploration and self-correction). Let $V(y) \in \{0, 1\}$ be a deterministic verifier. Our goal is to internalize verified deliberation into fast inference by maximizing the likelihood of parsimonious verified trajectories τ_{opt}: $\theta^* = \arg\max_\theta \mathbb{E}_{x\sim\mathcal{D}}\big[\log \pi_{\text{fast}}(\tau_{\text{opt}} \mid x)\big]$

4 Methodology

We propose Iterative Event-Driven Internalization (I-EDI), a closed-loop framework for iterative improvement of the the reasoning capacity. It transfers π_{slow} into the intuitive parameters of π_{fast}. As shown in Figure 2, I-EDI is a Spiral Curriculum. The framework runs in three phases: (1) Active Boundary Probing, (2) Verifiable Counterfactual Simulation, and (3) Parsimonious Internalization. Importantly, NbD is meant to test necessity of auditable actions under a fixed interface, not to penalize free-form reasoning. If the agent were allowed to introduce new tool calls after a deletion, the notion of necessity would quickly break down. Any removed node could be reconstructed through an alternative tool path, turning NbD into a test of search ability rather than causal dependence. Our runner therefore freezes the auditable interface and asks a sharper question: given the same observable tool outputs and the same allowed calls, is node u dispensable? The result is a conservative, precision-first core. That conservatism is deliberate—it prevents shortcut structures from slipping through and accumulating over successive self-evolution rounds.

4.1 Generative Frontier Probing

We implement G_ϕ as the same backbone LLM with an edit-operator prompt that applies one of: numeric substitution, isomorphic context rewrite, constraint perturbation while preserving the task type. We reject trivial edits (near-paraphrases or no-operator-change) and enforce hard constraints (e.g., keep the same answer format and required tool type). We sample n variants with temperature T, deduplicate them using normalized string hashes and an embedding similarity threshold ($< \epsilon$), and monitor diversity via the distribution of applied edit operators.

4.2 Verifiable Counterfactual Simulation (VCS)

For each frontier instance x (fast fails, slow succeeds), we first obtain a verified seed trace τ_{seed} and extract its Core Action Structure S: a minimal auditable action-dependency graph (not free-form text). We then generate M structure-preserving counterfactuals $\tilde{x}$ and run $\pi_{\text{slow}}(\tilde{x})$. A trace is admitted only if it passes: Execution $V(\tau) = 1$, TSC (its auditable graph is isomorphic to S under the same canonicalization), and NbD (deleting any core node flips verification under no-recompute replay). Finally, the admitted trace is projected to a parsimonious supervision target τ_{opt} to update π_{fast}.

ExtractCoreStructure(τ). We define the core S as the smallest auditable subgraph that is (i) sufficient to pass the verifier and (ii) necessary under our replay protocol (NbD). Concretely, starting from the full auditable graph $G(\tau)$, we iteratively remove candidate nodes and keep a deletion only if the replayed trace still verifies. The pruning order is fixed only to guarantee reproducibility, not to claim a unique mathematical optimum; when multiple deletion orders yield different minimal cores, we retain all verified minimal cores as an equivalence class $S(x)$ (below). Thus, verification never depends on one arbitrary template.

Algorithm 1 VCS: Core Structure Extraction & Verification

Require: Instance x', Policy π, Verifier V, Gen G_ϕ, Count M
Ensure: Dataset D

$\tau_0 \leftarrow \pi(x')$; **if** $V(\tau_0) = 0$ **then return** $\emptyset$
$S \leftarrow \textsc{GetCore}(\tau_0)$; $D \leftarrow \emptyset$; $\mathcal{N} \leftarrow \{G_\phi(x', \textsc{Preserve}(S))\}_{m=1}^{M}$
for $\tilde{x} \in \mathcal{N}$ **do**
  $\tau \leftarrow \pi(\tilde{x})$
  if $V(\tau) \neq 0$**and**$\textsc{Consistent}(\tau, S)$**and**$\textsc{NbD}(\tau, S, V)$ **then**
    $D \leftarrow D \cup \{(\tilde{x}, \textsc{Parsimonious}(\tau))\}$
  end if
end for
return D

TSC(τ, S). Extract $G(\tau)$ under the same canonicalization; accept iff there exists an isomorphism that matches all node signatures $(t, \text{name}, \text{args}^*, \text{outs}^*)$ and dependency edges. (QA uses set-equivalence for evidence chains.)

NbD(τ, S, V). For each $u \in S$, replay $\tau \setminus \{u\}$ under a deterministic **no-recompute runner**: (i) permit only the original auditable calls from τ excluding deleted nodes, in the same order; (ii) block any *new* auditable action not present in τ; (iii) additionally block any action a' whose canonical signature collides with u, where $\sigma(a) = (\texttt{name}, C(\texttt{args}), H(C(\texttt{outs})))$ and collision is $\sigma(a') = \sigma(u)$. Declare u necessary iff $V(\tau) = 1$ but $V(\tau \setminus \{u\}) = 0$. No-Recompute is an Attribution Test, not a Solvability Test. Allowing re-computation would merely verify if the problem can be solved (solvability), whereas NbD rigorously tests if the specific deleted node was statistically necessary for the current trajectory's success (causal attribution). This strict isolation is required to distill the exact winning logic.

Core Action Structure. Figure2 ("Joy reading book") illustrates S. A standard agent may reach the correct answer ("0.5 hours") via spurious cancellations (e.g., unnecessary unit conversions). **Definition:** we define S as the **minimal auditable dependency graph**, not the full reasoning text. For "Joy", S contains only: $Read\_Rate = 8/20 \rightarrow Total\_Time = 120/Read\_Rate \rightarrow Convert\_To\_Hours$. **Mechanism:** VCS (i) extracts an inclusion-minimal S by NbD pruning under the no-recompute replay, and (ii) stress-tests invariance with counterfactuals (e.g., "120 pages" $\rightarrow$ "200 pages"): a trace is admitted only if its auditable graph is **isomorphic** to S (TSC) and core deletions flip verification (NbD). Thus, answer-correct "lucky guesses" are rejected when their structure breaks under counterfactuals (TSC) or when deletion exposes hidden dependence (NbD); accepted traces are projected to the parsimonious target τ_{opt} to supervise π_{fast}.

Structure-Preserving Counterfactuals. A single frontier event is insufficient for robust internalization: the model must learn that surface form can change while the auditable dependency structure remains invariant. So we create a Counterfactual Contrast Set ($\mathcal{N}(x') = \tilde{x}_1, \ldots, \tilde{x}_M$) by changing the non-essential features (e.g., variable names, wording, distractors, or isomorphic context shifts),and we force the intended (S) to remain unchanged. Counterfactuals are intuitively as

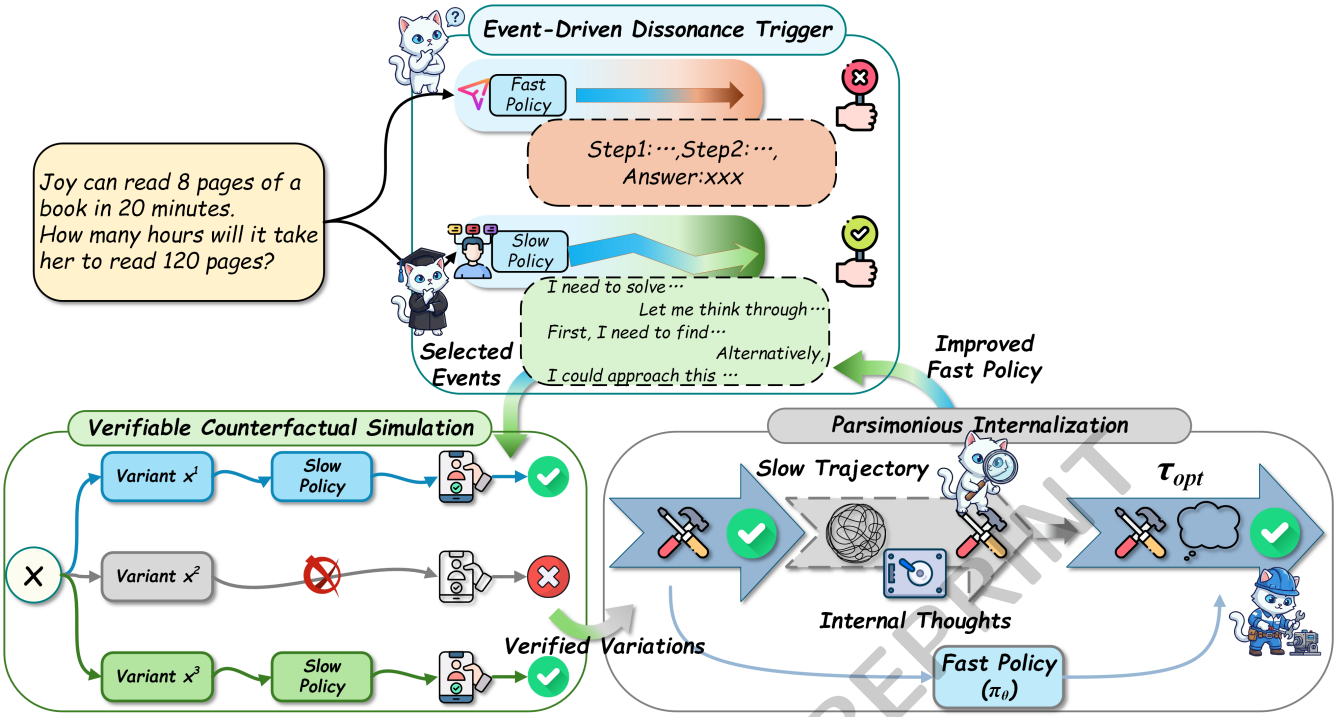

Figure 2: Overview of I-EDI.

"near-boundary drills" around the same basic decision structure.

Double Verification. A counterfactual candidate $\tilde{x} \in \mathcal{N}(x')$ is admitted only if its slow-policy trajectory $\tau_{\text{slow}}(\tilde{x})$ passes: (i) **Execution verification**: a deterministic verifier returns $V(\tau_{\text{slow}}(\tilde{x})) = 1$; (ii) **Structural verification**: the trace must be consistent with the extracted core structure S (TSC), and deleting any core node aligned with S must flip verification to failure (NbD). See Algorithm 1 (gates) and Table 1 (canonical signatures); the Appendix only lists additional edge cases and examples. We employ the strict 'no-recompute' constraint as a conservative precision-first. This intentionally trades recall for precision, ensuring that it can learn explicit dependencies rather than latent correlations or shortcuts. The resulting dataset ($\mathcal{D}^{(k)} *$ train) contains pairs ($(\tilde{x}, \tau *$ opt)), where (τ_{opt}) is the optimized tight trajectory derived from verified (τ_{slow}).

Given a slow-policy trace τ from π_{slow}, an auditable node is a typed record

$$u \triangleq \langle i,\ t,\ name,\ args^*,\ outs^*,\ meta \rangle, \tag{1}$$

where i is the step index; $t \in \{tool, retrieve, select, check, commit\}$; $name$ is the tool/API identifier; $args^* = \mathcal{C}(args)$ and $outs^* = \mathcal{C}(outs)$ are canonicalized arguments/outputs; and $meta$ stores optional hashes/timestamps/status. S is defined only over auditable actions (not free-form CoT), making deletion/necessity tests well-posed. Canonicalization $\mathcal{C}(\cdot)$ is *deterministic* and aims for *stable equivalence* when it is unambiguous; we use a conservative, locality-preserving design (normalize within each node's args/outs only, without modifying cross-node dependencies). We instantiate S across domains (Table 1): symbolic ops (Math) and retrieval/evidence APIs (QA).

Domain	Auditable Node Signature (u)	Canonicalization Rule $\mathcal{C}(\cdot)$
Math	**Tool:** symbolic ops (e.g., `simplify`); **Commit:** `commit(`$\hat{y}$`)`	**Term rewriting:** Parse w/ SymPy, serialize via `srepr`; sort operands; norm precision. Ambiguous → distinct.
QA†	**Retrieve:** $\langle q, \text{ids} \rangle$; **Select:** $\langle \text{d\_id}, \text{s\_id} \rangle$; **Commit:** `compose(`$\hat{y}$`, evs)`	**Grounded QA:** Sort doc IDs; evidence as $\langle \text{d\_id}, \text{s\_id} \rangle$; norm query whitespace. Set-equivalence for evidence.

Table 1: Instantiations of auditable nodes and canonicalization $C(\cdot)$. †Evidence-supervised QA only (e.g., HotpotQA).

Dependency Edge. For two nodes (u, v) add a directed edge $(u \to v)$ if (v)'s canonical arguments reference (u)'s canonical outputs:

$$u \to v \iff Ref(outs_u^*,\ args_v^*) = 1 \tag{2}$$

where (Ref) is a domain-specific reference detector (defined per domain below), implemented via ID linking, variable-binding tables, or hash references (preferred), and falls back to exact token match if necessary. The Core Action Structure

is a directed attributed graph:

$$S \triangleq (U,\ E,\ \psi) \tag{3}$$

where U is a subset of auditable action nodes extracted from τ, $E \subseteq U \times U$ are dependency edges, and ψ stores node attributes $(t, name, args*, outs*)$. We compute S by pruning non-essential nodes from the full auditable graph $G(\tau)$ using a verifier V. Informally, S contains the smallest set of nodes that are (i) sufficient to pass verification and (ii) necessary under deletion tests. We define $\mathcal{S}$ strictly over auditable actions rather than free-form natural-language steps. Table1 defines the auditable node signatures and canonicalization rules (per domain), and Algorithm 1 defines Ref/TSC/NbD operationally; the Appendix only adds extended rules and worked examples.

Deterministic (dataset-grounded) verification for multi-hop QA. Our deterministic/semi-deterministic QA claims are restricted to evidence-supervised multi-hop datasets (e.g., HotpotQA) with gold supporting facts; we do not claim deterministic verification for open-domain QA without such annotations. For QA, we instantiate S as an auditable evidence-chain graph $retrieve(query) \rightarrow select\_evidence(doc\_id, sent\_id) \rightarrow compose/commit$ (not free-form CoT), so TSC and NbD are executable. We define $V_{\text{QA}}(\tau) = 1$ iff the final answer matches and the selected evidence covers at least one gold supporting-fact set with threshold α, which is deterministic given dataset annotations and evidence IDs (Appendix for edge cases).

To enforce structural invariance across counterfactuals, we require the extracted dependency graph to match the seed structure $\mathcal{S}$ under canonicalization $\mathcal{C}(\cdot)$. Edges are defined by auditable data flow (Table 1): $u \rightarrow v$ iff $\text{Ref}(\text{outs}_u^*, \text{args}_v^*) = 1$. We accept a candidate trace only if it matches $\mathcal{S}$ exactly; for QA we relax strict isomorphism to *set-equivalence* over valid evidence chains to accommodate order invariance, filtering answer-correct but structurally spurious traces.

For a verified trajectory τ, NbD tests whether an auditable node u is operationally necessary under a fixed replay protocol: we replay $\tau \setminus u$ and declare u necessary iff the verifier flips, i.e.,$V(\tau) = 1$ but $V(\tau \setminus u) = 0$. For QA evidence-selection nodes, deleting u removes its $(doc\_id, sent\_id)$ tuple from the composed evidence; V_{QA} fails if either the answer is incorrect or the remaining evidence violates the supporting-fact check. Replay is enforced by a deterministic *no-recompute* runner to prevent recovery via alternative tool calls (details in Appendix). NbD deletes a node u and replays $\tau \setminus \{u\}$ under a conservative runner that prevents 're-deriving' deleted evidence via alternate tool paths. Concretely, during replay we (i) permit only the original auditable calls from τ excluding deleted nodes, in the same order, plus free-form text; (ii) block any newly proposed auditable action not present in τ; and (iii) additionally block any action a' whose canonical signature $\sigma(a')$ collides with the deleted node u, where $\sigma(a) := (\texttt{name}(a), C(\texttt{args}(a)), H(C(\texttt{outs}(a))))$ and collision is $\sigma(a') = \sigma(u)$ (exact match on name + normalized args; and, when outs exist, matching output-hash). This makes NbD a **precision-first necessity test**: if a solution still verifies under these constraints, the deleted node is not necessary; otherwise we conservatively keep it.

Multiple valid cores (budgeted). We maintain an equivalence class $S(x) = \{S_r\}_{r=1}^{R}$ of distinct minimal cores from trajectories that pass V/TSC/NbD. Cores are clustered by a canonical graph hash (node signatures + edges under the same canonicalization) and we keep top-R (default R=8) by frequency/parsimony. A trajectory is accepted if it matches **any** S_r (OR-over-cores); matching costs $O(R \cdot |E|)$ and Sec. 5 shows robustness to R.

Canonicalization robustness. When equivalence is ambiguous we keep structures distinct (precision-first); a strictness sweep in Appendix shows Iter3 is stable, with expected admit/match trade-offs and reported per-gate rejection rates.

4.3 Parsimonious Internalization

Naively cloning τ_{slow} makes π_{fast} imitate deliberation traces (trial-and-error) rather than internalize the underlying decision structure. We therefore train on a parsimonious trajectory that retains only auditable structure and the minimal bindings needed to execute it.

Parsimonious trajectory τ_{opt}. We deterministically construct $\tau_{\text{opt}} = \text{PROJ}(\tau_{\text{slow}}, S)$ by: (i) keeping all auditable actions in S (tool/retrieve/select/check/commit) in topological order; (ii) retaining only the minimal text spans that bind each action's canonical arguments (dropping alternatives and self-corrections); and (iii) keeping the final commit. In practice, we select the shortest contiguous spans that contain all symbols/IDs in the canonical argument set args* of actions in S (examples in Appendix).

Training. π_{fast} is updated by maximizing the likelihood of optimized trajectories:

$$\mathcal{L}_{\text{internalize}}(\theta) = -\sum_{(\tilde{x}, \tau_{\text{opt}}) \in \mathcal{D}_{\text{train}}^{(k)}} \sum_{t=1}^{|\tau_{\text{opt}}|} \log \pi_\theta(a_t | \tilde{x}, a_{<t}). \tag{4}$$

Training on τ_{opt} teaches π_{fast} to execute verified cores without reproducing the slow scaffolding. We iterate until validation performance plateaus; as π_{fast} absorbs the current frontier, Active Probing naturally shifts to harder instances, yielding a spiral curriculum from novice to expert.

4.4 Experimental Setup

Implementation Details. We use Qwen3-4B-Base and Qwen3-8B-Base as the backbone. The intuitive policy π_{fast} uses greedy decoding (T=0), while the slow teacher π_{slow} follows a Plan–Execute–Critique–Repair protocol with T=0.7. We run K=3 evolution rounds; in each round, candidates must pass VCS (execution correctness and TSC) and only verified traces are used for NLL fine-tuning. Key hyperparameters are n=16, T_{probe}=0.7, ε=0.92, M=8, R=8, and WL-iters= 2. Unless stated otherwise, each seed reruns the full pipeline (probing $\rightarrow$ VCS filtering $\rightarrow$ fine-tuning). The Appendix reports a light sensitivity sweep over $(M, R, \varepsilon, T_{\text{probe}})$, showing stable Iter3 performance with predictable gate-rate trade-offs.

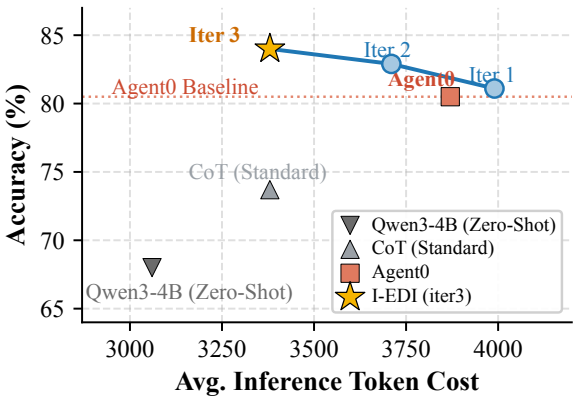

Figure 3: Accuracy vs. inference token cost on MATH across evolution rounds; later rounds shift upward-left.

Model Name	Minerva	MATH	GSM8K	Olympiad
Qwen3-4B-Base				
Base	39.6	71.0	88.6	43.7
+ Absolute Zero	41.9	76.2	89.3	41.5
+ SPIRAL	42.4	76.4	91.0	38.4
+ R-Zero	52.9	79.6	92.1	44.6
+ Agent0	55.6	80.5	92.6	46.7
+ I-EDI(*iter1*)	56.2	81.1	92.4	47.8
+ I-EDI(*iter2*)	57.8	82.9	93.6	48.6
+ I-EDI(*iter3*)	**58.6**	**84.0**	**95.1**	**50.5**
Qwen3-8B-Base				
Base	54.9	79.2	90.7	47.9
+ Absolute Zero	52.9	76.6	92.0	47.8
+ R-Zero	60.7	82.0	94.1	48.9
+ Socratic-Zero	52.4	81.2	87.3	55.1
+ Agent0	61.3	82.4	94.5	54.0
+ I-EDI(*iter1*)	61.0	81.2	92.5	52.1
+ I-EDI(*iter2*)	62.8	83.1	95.4	54.8
+ I-EDI(*iter3*)	**64.6**	**84.2**	**97.0**	**55.5**

Table 2: Results on mathematical reasoning benchmarks (greedy decoding). Bold = training peak.

Benchmarks and Baselines. We use datasets that have deterministic verifiers (GSM8K[Cobbe *et al.*, 2021], Minerva[Lewkowycz *et al.*, 2022], MATH[Hendrycks *et al.*, 2021] and the math subset of Olympiad-Bench[He *et al.*, 2024] and HotpotQA[Yang *et al.*, 2018]) for seed generation. For evaluation, we also run out-of-domain benchmarks with no training exposure, including SuperGPQA[Du *et al.*, 2025], MMLU-Pro[Wang *et al.*, 2024], and BBEH[Kazemi *et al.*, 2025]. We compare against standard base models and strong self-evolving agents: R-Zero[Huang *et al.*, 2025], Absolute Zero[Zhao *et al.*, 2025], SPIRAL[Liu *et al.*, 2025], Socratic-Zero[Wang *et al.*, 2025a], and Agent0[Xia *et al.*, 2025] to keep controlled inference budgets for all methods.

Decontamination. To prevent data leakage, we do a strict double-stage cleaning filter (exact hashing and semantic similarity checks) to remove the generated instances that align with test sets.

4.5 Main Result

We present results for mathematical reasoning and general-domain reasoning in Table 2 and Table 3.

Comparison with Baselines. I-EDI consistently outperforms strong baselines across model sizes. Notably, I-EDI (Iter 3) significantly surpasses the tool-integrated Agent0 on Qwen3-8B-Base. Its clear lead on Olympiad-Bench demonstrates robust complex problem-solving capability without test-time sampling. Furthermore, the performance gain from Iter 1 to 3 validates our evolutionary loop: active frontier internalization steadily enhances reasoning robustness.

Generalization. Table 3 highlights impressive out-of-domain results, where I-EDI achieves superior average scores on symbolic reasoning benchmarks. This confirms that I-EDI learns transferable, structure-grounded behaviors rather than task-specific memorization, enabling robust generalization. Figure 3 plots accuracy-token trajectories across iterations for both Agent0 and I-EDI under matched evaluation (greedy decoding). Early iterations may increase average tokens due to exploration, while later iterations achieve the intended upward-left shift as π_{fast} internalizes verified structure and relies less on verbose deliberation.

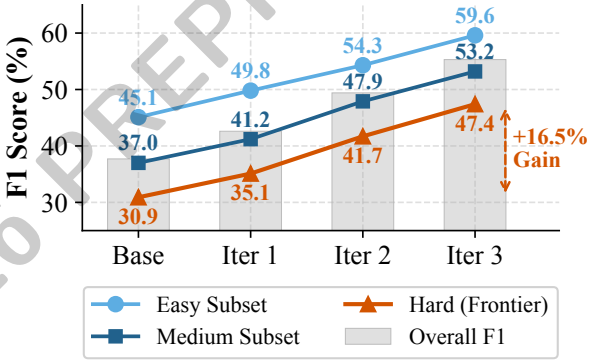

Figure 4: Capability Expansion on HotpotQA.

4.6 Capability Frontier Expansion on HotpotQA

We investigate whether I-EDI gives rise to genuine capability expansion instead of simply distorting existing reasoning patterns with Qwen3-4B. As shown in Figure 4, I-EDI has achieved a significant improvement in all difficulty buckets of HotpotQA across multiple evolution rounds. The improvement performance on Hard instances is close to the overall improvement, which indicates that iterative internalization process helps to gradually expand the agent's ability to tackle more complex multi-hop reasoning problems, rather than causing it to overfit on simpler instances.

4.7 Ablation Study: Stability of Iterative Self-Evolution

Table 4 shows that only the complete I-EDI pipeline achieves consistent gains across three evolution rounds on MATH (Qwen3-4B), while the ablation studies show either saturation or degradation in the later rounds. Specifically: (i) w/o Active Boundary Probing quickly saturates, highlighting the necessity of frontier mining for targeted capability growth; (ii) w/o Structural Verification reveals the common failure mode in self-training—initial gains that are followed

Model Name	SuperGPQA	MMLU-Pro	BBEH
Qwen3-4B-Base			
Base	25.8	42.9	8.32
+ Absolute Zero	27.1	52.6	8.3
+ SPIRAL	27.1	53.2	9.57
+ R-Zero	27.6	51.5	10.4
+ Agent0	29.9	55.9	12.0
+ I-EDI(*iter1*)	30.4	56.4	12.5
+ I-EDI(*iter2*)	30.9	56.9	13.1
+ I-EDI(*iter3*)	**31.5**	**57.3**	**13.4**
Qwen3-8B-Base			
Base	29.5	54.8	9.37
+ Absolute Zero	33.5	62.5	10.8
+ R-Zero	31.4	58.2	10.6
+ Socratic-Zero	30.1	60.9	9.5
+ Agent0	33.0	63.4	13.7
+ I-EDI(*iter1*)	33.7	63.9	14.2
+ I-EDI(*iter2*)	34.6	64.3	14.8
+ I-EDI(*iter3*)	**35.1**	**65.0**	**15.1**

Table 3: Results on general-domain reasoning benchmarks.

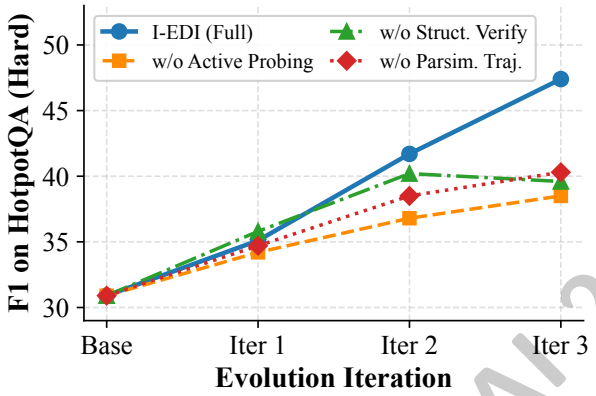

Figure 5: Hard-only capability frontier evolution on HotpotQA.

by a decline—indicating that simply verifying the final answers isn't enough, as shortcut trajectories tend to accumulate over rounds; (iii) w/o Parsimonious Internalization (training on $\tau_s low$ instead of $\tau_o pt$) reduces consolidation to imitation of deliberation artifacts, yielding weaker and less stable improvements. These findings bolster our argument that stable self-evolution requires a combination of frontier targeting, structure-preserving verification, and parsimonious internalization; the low variance is consistent with deterministic gating and fixed evaluation. Figure 5 confirms the same trend on HotpotQA-Hard: only I-EDI sustains steady growth; ablations either stagnate (no probing), regress late (no structural verification), or progress slowly (raw traces).

Figure 6 reports per-iteration generation–verification accounting: candidates shrink as π_{fast} improves (fewer frontier cases survive), while the verified-acceptance ratio rises—indicating cleaner data rather than filter overfitting, since the frontier rate and edit-operator entropy stay high across rounds.

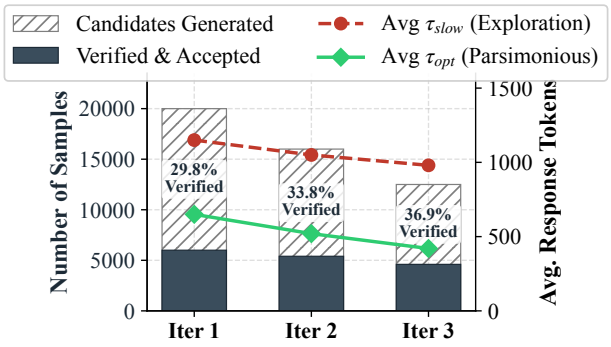

Figure 6: I-EDI generation–verification accounting (per iteration).

Variant	Iter1	Iter2	Iter3
I-EDI (Full)	81.1±0.21	82.9±0.18	**84.0±0.17**
w/o Active Boundary Probing	81.0±0.29	81.4±0.31	81.6±0.34
w/o Structural Verification	81.2±0.26	**82.0±0.28**	81.7±0.33
w/o Counterfactual Generation	80.8±0.30	81.0±0.32	81.3±0.35
w/o Parsimonious Trajectory (τ_{opt})	**81.3±0.24**	81.7±0.27	82.0±0.29
Static Distillation	81.1±0.33	81.0±0.35	80.9±0.37

Table 4: Ablation study on the stability of iterative self-evolution. Numbers are reported as mean±std over 5 random seeds, where each seed re-runs data generation (probing + VCS filtering) and fine-tuning. No-recompute: violations 56% deterministically blocked; stable NbD/verifier 96%.

Tp	Transformation	Purpose
A	**Param shift** (5 tosses/2 heads → 4/3)	Prevent memorization; enforce re-instantiation.
B	**Context switch** (coins → 6 kids/2 girls)	Test isomorphic transfer across domains.
C	**Edge case** (exactly 0 heads)	Probe boundary robustness & consistency.

Table 5: VCS counterfactual variants for a binomial-probability seed.

5 Conclusion

We presented I-EDI, a self-evolution framework that turns verified deliberation into compact parametric intuition by evolving on cognitive-dissonance events and enforcing anti-shortcut learning via verifiable counterfactual simulation. This yields stable capability growth across iterations, with pronounced gains on frontier instances and reduced redundant reasoning. Our current instantiations require auditable actions with deterministic or dataset-grounded verification, covering symbolic math and evidence-supervised multi-hop QA; extending structural verification beyond verifiable domains is future work.

Acknowledgments

This work is partly supported by the Project of Beijing Welinkirt DaoAI Tech-nologies Co., Ltd.

References

[An *et al.*, 2024] Shengnan An, Zexiong Ma, Siqi Cai, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Can llms learn from mistakes? an empirical study on reasoning tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 833–854, 2024.

[Chen, 2021] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[Cobbe *et al.*, 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[Do *et al.*, 2025] Cong Thanh Do, Rama Sanand Doddipatla, and Kate Knill. Effectiveness of chain-of-thought in distilling reasoning capability from large language models. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 833–845, 2025.

[Du *et al.*, 2025] Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.

[Gao *et al.*, 2025] Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 1, 2025.

[He *et al.*, 2024] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, 2024.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[Huang *et al.*, 2025] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

[Kazemi *et al.*, 2025] Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, et al. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26473–26501, 2025.

[Lee *et al.*, 2024] Hojae Lee, Junho Kim, and SangKeun Lee. Mentor-kd: Making small language models better multi-step reasoners. *arXiv preprint arXiv:2410.09037*, 2024.

[Lewkowycz *et al.*, 2022] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

[Liang *et al.*, 2025] Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, et al. Sage: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, page 130470, 2025.

[Lin *et al.*, 2025] Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Kun Wang, et al. Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. *arXiv preprint arXiv:2509.18970*, 2025.

[Liu *et al.*, 2025] Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, et al. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *arXiv preprint arXiv:2506.24119*, 2025.

[Lu *et al.*, 2023] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.

[Luo *et al.*, 2025] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

[Pryzant *et al.*, 2023] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

[Qian *et al.*, 2024] Chen Qian, Jiahao Li, Yufan Dang, Wei Liu, YiFei Wang, Zihao Xie, Weize Chen, Cheng Yang, Yingli Zhang, Zhiyuan Liu, et al. Iterative experience refinement of software-developing agents. *arXiv preprint arXiv:2405.04219*, 2024.

[Qiu *et al.*, 2025] Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

[Rein *et al.*, 2024] 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. In *First Conference on Language Modeling*, 2024.

[Shinn *et al.*, 2023] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[Sumers *et al.*, 2024] Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents, 2024.

[Team *et al.*, 2023] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

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

[Wang *et al.*, 2023b] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*, 2023.

[Wang *et al.*, 2024] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[Wang *et al.*, 2025a] Shaobo Wang, Zhengbo Jiao, Zifan Zhang, Yilang Peng, Xu Ze, Boyu Yang, Wei Wang, Hu Wei, and Linfeng Zhang. Socratic-zero: Bootstrapping reasoning via data-free agent co-evolution. *arXiv preprint arXiv:2509.24726*, 2025.

[Wang *et al.*, 2025b] Wei Wang, Zhaowei Li, Qi Xu, Yiqing Cai, Hang Song, Qi Qi, Ran Zhou, Zhida Huang, Tao Wang, and Li Xiao. Qcrd: Quality-guided contrastive rationale distillation for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14345–14356, 2025.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[Wu *et al.*, 2024] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456*, 2024.

[Wu *et al.*, 2025a] Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. *arXiv preprint arXiv:2502.04644*, 2025.

[Wu *et al.*, 2025b] Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

[Xia *et al.*, 2025] Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

[Yang *et al.*, 2018] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

[Yang *et al.*, 2025a] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[Yang *et al.*, 2025b] Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, et al. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *arXiv preprint arXiv:2510.08002*, 2025.

[Zhao *et al.*, 2025] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

[Zhou *et al.*, 2022] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.