WHEN UPDATING STOPS BEING LEARNING: RETHINKING LLM SELF-EVOLUTION VIA LEARNABLE INFORMATION GAIN

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolution lets large language models (LLMs) improve iteratively using their own generated data, but often suffers from self-evolution degeneration: performance improves, plateaus, then declines. Existing methods address this issue at the component level, targeting either the Questioner or the Solver, and overlook that self-evolution is a tightly coupled system. We propose a holistic framework based on learnable information gain, which measures how much novel, parameterizable information a round provides relative to the previous round. Theoretically, this gain equals the Kullback–Leibler divergence between the two rounds' data distributions plus their entropy change. Practically, it is estimated by fitting a small language model to the previous round and scoring new data via negative log-likelihood. Based on this diagnostic, we propose ATRI (Adaptive Training Regulation via Information-gain), which reweights samples within a round and halts training across rounds when information gain remains low. Experiments on popular datasets demonstrate the superiority of our proposal. Code is available at `https://anonymous.4open.science/r/ATRI-01C2`.

1 INTRODUCTION

Large language models have achieved significant progress in a range of tasks, yet they still depend on high-quality training data, which is becoming increasingly scarce and costly (Chen et al., 2024; Yuan et al., 2024). To address this, researchers have recently turned to *self-evolution*, a paradigm that enables the model to iteratively improve itself without external supervision, and one that has proved effective in mathematical reasoning, code generation, and multi-step agent tasks (Zelikman et al., 2022; Yuan et al., 2024; Chen et al., 2024; Wang et al., 2025; Huang et al., 2025b; Zhao et al., 2025). Self-evolution generally follows a multi-round training paradigm. In each round, a Questioner performs *Question Generation* and *Question Selection* to build the round's question set, and the Solver then performs *Answer Generation* and *Answer Evaluation* to produce and score candidate answers. Finally, the Questioner and Solver are iteratively updated through supervised fine-tuning (Zelikman et al., 2022; Chen et al., 2024) or reward-based reinforcement learning (Wang et al., 2025).

Despite the appeal of self-evolution approaches, they commonly encounter the challenge of *self-evolution degeneration*, a phenomenon in which solver performance improves during early rounds, then plateaus, and ultimately degrades (Shumailov et al., 2024; Alemohammad et al., 2024; Wang et al., 2025; Huang et al., 2025b) (Figure 1, left). Existing work addresses this issue from two complementary directions (Appendix A). The first direction seeks to explain the underlying mechanism: verifier-based accounts attribute it to reward hacking (Khalaf et al., 2025), data-based accounts to distributional or diversity collapse (Shumailov et al., 2024; Li et al., 2026), and gap-based accounts to the closing of the generation–verification gap (Song et al., 2024). The second direction focuses on mitigation: filtering-based methods select higher-quality training samples by leveraging reward variance or gradient statistics (Wang et al., 2025), while regularization-based methods impose diversity penalties to prevent the generated data from becoming overly homogeneous (Li et al., 2026).

Despite this progress, existing methods that alleviate performance degradation generally diagnose and intervene from only a partial perspective, focusing solely on either the Questioner or the Solver (Wang et al., 2025; Cui et al., 2025; Li et al., 2026), overlooking the fact that the Questioner–Solver system evolves as a coupled whole. For instance, reward-variance filtering operates exclusively at the *Answer*

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

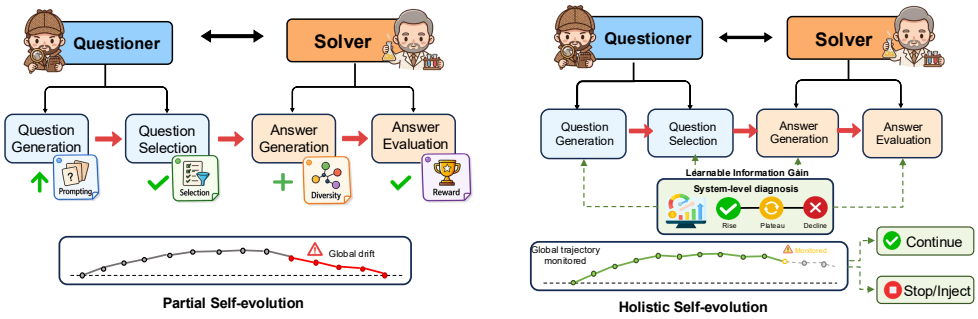


Figure 1: Previous self-evolution loops monitor each component separately and drift into decline (left). Our ATRI model tracks system-level diagnostic (right).

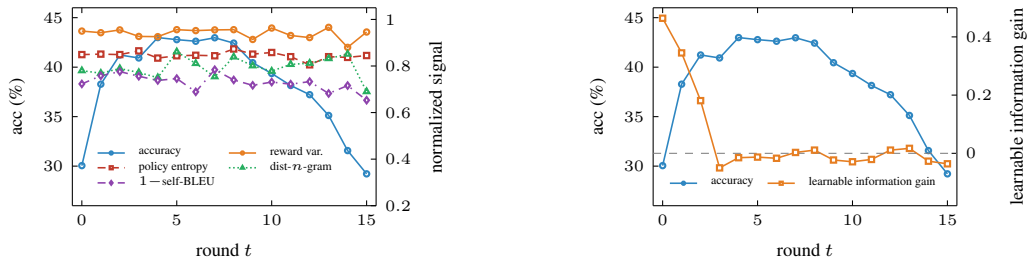
Evaluation stage to mitigate reward hacking by discarding low-variance samples (Wang et al., 2025). However, upstream stages such as *Answer Generation* may already have undergone corruption and degeneration due to the reduced diversity of training data, ultimately limiting the effectiveness of such downstream mitigation strategies. More critically, remediation applied at one training stage may inadvertently exacerbate degeneration at other stages (Wu et al., 2024). The underlying cause is that self-evolution constitutes an interconnected system in which each component is tightly coupled to the others (Sun et al., 2026). Such localized interventions targeting a single component may be insufficient to redirect the overall optimization trajectory (Yue et al., 2025; Wu et al., 2024).

Different from existing approaches that focus on a component-level perspective, we propose to understand and mitigate self-evolution degradation from a holistic, system-level perspective grounded in first principles. As illustrated in Figure 1, our approach consolidates signals throughout the entire training pipeline to enable holistic understanding and remediation. The nucleus of self-evolution lies in improving LLMs by extracting valuable information from their own outputs, which is essentially an information reprocessing process. To characterize this process quantitatively, we define the **learnable information gain** as the amount of extracted information that can be parameterized into the model and contributes to improving its performance. It theoretically quantifies how much novel content a training round's data contains compared with the previous round, being positive when new information appears and zero when the two rounds convey identical content. Tracking this gain across rounds thus reveals how much the model can still learn from its own outputs, allowing us to observe self-evolution as it approaches saturation and to forecast degradation before it occurs.

Concretely, we fit a probability distribution to the data generated at $(t-1)$-th round of self-evolution and use it to score the data from rounds $(t-1)$ and t via negative log-likelihood (NLL). The learnable information gain is defined as the extent to which round t's data scores worse (i.e., incurs higher NLL) than $(t-1)$-th round's data under this fitted distribution. The underlying intuition is as follows: the NLL of a sample with respect to a given distribution provides a precise measure of the information that this distribution fails to capture. Consequently, if the data from the new round remains poorly predicted by a model fit to the previous round, this indicates the presence of content that has not yet been learned. Theoretically, we show that this quantity admits a clean decomposition: it equals the KL divergence between the two rounds' data distributions plus their change in entropy under an exact fit, which endows it with a principled information-theoretic interpretation. In practice, the fitted distribution is implemented as a small language model retrained at each round. Building on this notion, we propose ATRI (Adaptive Training Regulation via Information-gain), a lightweight component designed to alleviate the challenge of degeneration. ATRI operates at two levels: *within* a training round, each sample is reweighted according to how far its information-gain score exceeds the previous round's average, thereby concentrating gradient updates on novel content; *across* rounds, ATRI halts training once the information gain remains below a small threshold for two consecutive rounds, preempting severe degeneration before it takes hold. We extensively evaluate our proposal on several popular datasets, and the experimental results demonstrate its superiority.

We make the following three major contributions:

- **A system-level diagnostic for self-evolution degeneration.** We introduce the *learnable information gain*, prove that, under an exact fit, it decomposes into the KL divergence between rounds plus their entropy change, and use it to explain the mechanism underlying self-evolution degeneration.



(a) Partial signals vs. accuracy

(b) Information gain vs. accuracy

Figure 2: Analysis on Qwen3-4B-Base with MATH. (a) Partial signals cannot reflect performance degradation. (b) Lower learnable information gain can signal later performance degradation.

- **An information-gain-driven paradigm.** We propose ATRI, which reweights samples toward novel content within each round and halts self-evolution once the gain remains low across consecutive rounds, effectively mitigating degeneration.
- **Extensive experimental evaluation.** We evaluate our approach on several reasoning datasets and demonstrate that it can forecast degeneration before it occurs and alleviate degeneration.

2 Preliminary Analysis

Prior work monitors training degeneration via signals such as reward variance (Wang et al., 2025), policy entropy (Cui et al., 2025), or output diversity (Li et al., 2016; Zhu et al., 2018). However, each signal captures only part of the self-evolution loop: reward variance reflects the answer evaluation module, policy entropy reflects the model policy, and diversity reflects only the generated outputs. To demonstrate that such partial signals are incapable of reliably tracking performance degeneration, we design the following experiment based on a popular model, R-Zero (Huang et al., 2025b). R-Zero is trained on Qwen3-4B-Base with the MATH dataset (Hendrycks et al., 2021) for 15 rounds, following the original settings. A generated answer is counted as correct only if it exactly matches the ground-truth answer. At each round, we record held-out accuracy as the performance measure (Wang et al., 2025), along with four partial signals: reward variance (Wang et al., 2025), policy entropy (Cui et al., 2025), distinct-n-gram ratio (Li et al., 2016), and $1 -$ self-BLEU (Zhu et al., 2018). For plotting, all four partial signals are normalized across rounds.

In Figure 2a, accuracy peaks at round 4 and declines through round 15, while the four partial signals fluctuate without signaling degeneration. In contrast, C_t reaches zero at round 3 and then fluctuates around it (Figure 2b), anticipating the peak and tracking the loss of new learnable information.

3 Methodology

Figure 3 provides an overview of ATRI. In each round, the Questioner is updated first, followed by the Solver, following the self-evolution loop described in Section 1. ATRI keeps this loop and inserts one module into both trainings. The module computes the learnable information gain. It fits a proxy model on the previous round, scores the current round, and turns the scores into sample weights. It reweights the Questioner update (Section 3.1), reweights the Solver update (Section 3.2), and determines when to stop (Section 3.3).

3.1 Questioner Training with the Learnable Information Gain

This subsection describes the training paradigm of the Questioner guided by the proposed learnable information gain. First, we present its general training procedure. We then explain how to estimate the information gain and use it to facilitate training.

3.1.1 General Training Reward for Questioner

The general reward of Questioner is following R-Zero (Huang et al., 2025b). Specifically, the Questioner Q_θ first generates a batch of B questions $\{q_i\}_{i=1}^{B}$ from a fixed prompt p_0. A useful question should be hard yet solvable for the current Solver S_ϕ. For each q_i, the Solver generates m answers, the most frequent answer is taken as the pseudo-label $\tilde{y}_i$, and $\hat{p}_i$ denotes the fraction of the m answers that agree with $\tilde{y}_i$. Since $\hat{p}_i$ approximates the Solver's success probability on q_i, the resulting *uncertainty reward* is designed to peak when the Solver is correct about half of the time,

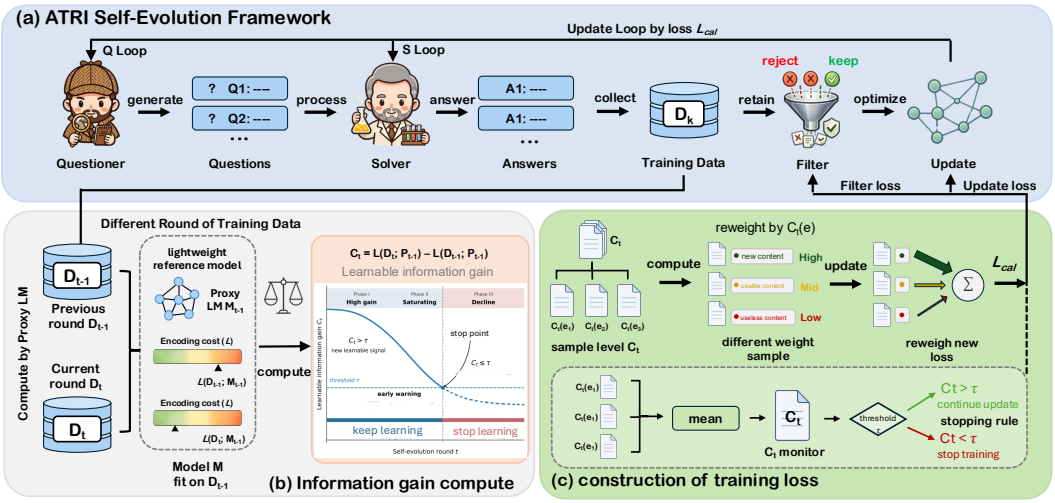


Figure 3: The overview of the proposed ATRI framework.

i.e., when the question is maximally uncertain:

$$r_{\text{unc}}(q_i) = 1 - 2\left|\hat{p}_i - \tfrac{1}{2}\right|. \tag{1}$$

Optimizing r_{unc} alone, however, can collapse the batch toward near-duplicate questions at the same difficulty level, so a redundancy penalty is imposed within the batch. Questions are clustered by BLEU similarity. For a question q_i in cluster $\mathcal{C}_k$, the *repetition penalty* is

$$r_{\text{rep}}(q_i) = \lambda|\mathcal{C}_k|/B. \tag{2}$$

Combining the two terms and zeroing out questions that fail the format check, the overall reward is

$$r_Q(q_i) = \max\big(0,\ r_{\text{unc}}(q_i) - r_{\text{rep}}(q_i)\big). \tag{3}$$

The reward is then normalized within the batch into the GRPO advantage (Shao et al., 2024),

$$\hat{A}_i = \frac{r_Q(q_i) - \text{mean}\big(r_Q(q_1), \ldots, r_Q(q_B)\big)}{\text{std}\big(r_Q(q_1), \ldots, r_Q(q_B)\big) + \epsilon}, \tag{4}$$

where $\text{mean}(\cdot)$ and $\text{std}(\cdot)$ are the batch-wise mean and standard deviation, respectively, yielding $\hat{A}_i$, which trains Q_θ with the standard GRPO loss.

3.1.2 Learnable Information Gain Estimation

The general reward only offers heuristic guidance within a single training round. It cannot measure whether the generated questions carry valuable, learnable information for the next round's evolution. To address this, ATRI uses a proxy model to estimate information gain, which measures how much the current round departs from the previous one.

Proxy Model Training. Consider the self-evolution loop after round $t-1$. The Questioner and Solver yield a set of question–answer pairs $\mathcal{D}_{t-1} = \big\{(q_{t-1}^{(i)},\ s_{t-1}^{(i)})\big\}_{i=1}^{N_{t-1}}$, which serves as the training data for the next round. We seek a compact and queryable summary of the round's outputs, so that different rounds can be compared on a common footing.

To this end, we fit a small language model M_{t-1} on $\mathcal{D}_{t-1}$ and refer to it as the *proxy*. Two design choices are deliberate. First, the proxy is not to solve the downstream task but to provide a low-variance estimate of the round's data distribution. Second, the proxy models the text of the round rather than the question-to-answer mapping. Each pair is concatenated into a single sequence $x = [q_{t-1}; s_{t-1}]$ and treated as unstructured text, so that the phrasing of a question and the form of its solution are characterized jointly. For a sequence x with tokens $x_1, \ldots, x_{|x|}$, the proxy scores it as

$$\ell(x) = -\frac{1}{|x|}\sum_{k=1}^{|x|} \log M_{t-1}(x_k \mid x_{<k})\,, \tag{5}$$

i.e., its negative log-likelihood per token. Length normalization removes the trivial dependence on sequence length, making $\ell(x)$ comparable across pairs of differing verbosity: $\ell(x)$ is small when x is typical of round $t-1$ and large when x is atypical of round $t-1$. The proxy is itself trained with the

standard autoregressive language-modeling objective:

$$\mathcal{L}_{\text{proxy}}(M_{t-1}) = \frac{1}{|\mathcal{D}_{t-1}|} \sum_{x \in \mathcal{D}_{t-1}} \ell(x) = -\frac{1}{|\mathcal{D}_{t-1}|} \sum_{x \in \mathcal{D}_{t-1}} \frac{1}{|x|} \sum_{k=1}^{|x|} \log M_{t-1}\big(x_k \mid x_{<k}\big). \quad (6)$$

The negative log-probability of a token is the difficulty of predicting it, and training drives this difficulty down on $\mathcal{D}_{t-1}$. Because this reduction is shared across the many sequences in $\mathcal{D}_{t-1}$, it is the recurring patterns of the round that are learned, rather than any single sequence in isolation. After training, M_{t-1} therefore predicts the patterns of round $t-1$ more easily, while text falling outside these patterns remains difficult to predict. $\ell(x)$ thus measures how far a text lies from what the previous round contains, and we adopt it as the measurement of information.

Learnable Information Gain of a Question. Updating the Questioner requires assigning a scalar score to each generated question that reflects the amount of new information it contributes. Since the proxy model can evaluate the question component of a (q, s) pair in isolation, we obtain a raw cost $\ell(q)$ for each question. However, this cost is not informative in absolute terms: it should be interpreted relative to a baseline. We therefore compare $\ell(q)$ with the previous-round mean difficulty,

$$\bar{\ell}_q(\mathcal{D}_{t-1}) = \frac{1}{|\mathcal{D}_{t-1}|} \sum_{(q',s') \in \mathcal{D}_{t-1}} \ell(q'), \quad (7)$$

which serves as a reference point for what the model already knows how to ask. The learnable information gain of a question q is then defined as the deviation from this baseline:

$$c_t^Q(q) = \ell(q) - \bar{\ell}_q(\mathcal{D}_{t-1}). \quad (8)$$

Intuitively, $c_t^Q(q) > 0$ indicates that q has a larger prediction difficulty than the average question in the previous round, and thus likely elicits content beyond what earlier questions already covered.

3.1.3 Training Objective Function

Having defined c_t^Q, we now integrate it into the Questioner update introduced in Section 3.1. The key idea is that questions with positive c_t^Q carry information not already present in the previous round, and the update should therefore be concentrated on them. Since c_t^Q can be negative, we take its positive part as the weight of a question,

$$w_t^Q(q_i) = \max\big(c_t^Q(q_i),\, 0\big). \quad (9)$$

This weight rescales the advantage term in Equation (4), yielding a calibrated advantage

$$\tilde{A}_i = w_t^Q(q_i)\, \hat{A}_i, \quad (10)$$

which replaces $\hat{A}_i$ in the standard GRPO objective to give the Questioner loss of ATRI,

$$\mathcal{L}^Q(\theta) = -\frac{1}{B} \sum_{i=1}^{B} \tilde{A}_i \log Q_\theta(q_i \mid p_0) + \beta\, \mathrm{KL}\big(Q_\theta \,\|\, Q_{\text{ref}}\big), \quad (11)$$

where B is the batch size, as in Equation (4), Q_{ref} denotes the Questioner at the start of the round, and we omit the GRPO clipping terms for clarity.

3.2 Solver Training with the Learnable Information Gain

The Solver follows the same pattern as the Questioner. After the Questioner update, the Solver S_ϕ is trained on questions proposed by the updated Questioner. The Questioner samples a pool of candidate questions, and for each candidate q_i the Solver samples m answers; the pseudo-label $\tilde{y}_i$ and agreement fraction $\hat{p}_i$ are obtained as in Section 3.1. Following the standard filtering strategy of Questioner–Solver methods (Huang et al., 2025b; Li et al., 2026), a candidate question is retained only if $|\hat{p}_i - \frac{1}{2}| \le \delta$, which discards questions that are too easy or too hard for the current Solver. The Solver then generates a fresh group of answers for each retained question, and these question-answer pairs form the training set $\mathcal{D}_t$ of round t. Each answer s_j to a retained question q_i receives the binary reward $r_S(s_j \mid q_i) = \mathbb{1}[s_j = \tilde{y}_i]$, which is normalized within its group into the advantage $\hat{A}_j$ as in Equation (4). The standard update trains S_ϕ with the GRPO loss using $\hat{A}_j$.

The same calibration module then applies on the answer side. The proxy scores each answer conditioned on its question, giving a raw prediction difficulty $\ell(s_j \mid q_i)$, and its learnable information gain is defined analogously to Equation (8) as the deviation from the previous-round mean cost:

$$c_t^S(s_j \mid q_i) = \ell(s_j \mid q_i) - \bar{\ell}_{s|q}(\mathcal{D}_{t-1}), \quad (12)$$

where $\bar{\ell}_{s|q}(\mathcal{D}_{t-1})$ denotes the average answer cost of the previous round. As on the Questioner side, we take the positive part $w_t^S(s_j \mid q_i) = \max\big(c_t^S(s_j \mid q_i),\, 0\big)$ to rescale the advantage,

$$\tilde{A}_j = w_t^S(s_j \mid q_i)\, \hat{A}_j, \tag{13}$$

which yields the ATRI Solver objective, with S_{ref} denoting the round-start Solver:

$$\mathcal{L}^S(\phi) = -\frac{1}{|\mathcal{D}_t|} \sum_{(q_i, s_j) \in \mathcal{D}_t} \tilde{A}_j \log S_\phi(s_j \mid q_i) + \beta\, \mathrm{KL}\big(S_\phi \,\|\, S_{\mathrm{ref}}\big), \tag{14}$$

Appendix E presents the derivation for the shared-parameter setting.

3.3 Iterative Co-Evolution Paradigm with Early Warning

Sections 3.1 and 3.2 together specify the two updates that constitute one round of ATRI. Concretely, one round proceeds as follows: the proxy M_{t-1} is trained on $\mathcal{D}_{t-1}$; the Questioner generates its questions and is updated with $\mathcal{L}^Q$; the Solver then forms $\mathcal{D}_t$ and is updated with $\mathcal{L}^S$; and the next round begins from the resulting Questioner and Solver. A natural question is when to stop this process: intuitively, rounds should stop once a new round no longer adds anything beyond the previous one. Answering this requires a score for the whole round, which the proxy model is capable of providing.

Learnable Information Gain of a Round. Thus far the proxy model has scored either the question part or the answer part of a pair. For a full round, we instead score each complete pair using its full text $x = [q; s]$. The cost of a round is the average cost over its pairs, $\bar{\ell}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ell(x)$. As with a single question in Equation (8), we compare this cost against that of the previous round under the same proxy model, defining the learnable information gain of round t as

$$C_t = \bar{\ell}(\mathcal{D}_t) - \bar{\ell}(\mathcal{D}_{t-1}). \tag{15}$$

C_t measures the additional cost of the current round relative to the previous one. A positive C_t suggests that the current round contains patterns not captured in the previous round, while a value close to zero indicates that little additional content is introduced and that further rounds may be unnecessary. Since each pair contains both a question and an answer, C_t can be decomposed into c_t^Q and c_t^S. Appendix C provides the corresponding question and answer terms. Appendix I shows that, under an exact fit, it equals the KL divergence between rounds plus the entropy change.

Three-phase Lifecycle. Across rounds, C_t tends to decrease, since what is new at round t is absorbed into the previous round by round $t+1$ (Proposition 3). This decreasing trend naturally divides self-evolution into three phases. In Phase I, C_t is positive and accuracy rises. In Phase II, C_t approaches zero and accuracy plateaus. In Phase III, C_t remains at zero while training continues; the model keeps fitting samples it can already generate, and accuracy degrades. Proposition 6 shows that this phase behaves like distilling the model onto its own high-confidence outputs. The vanilla model run of Section 2 passes through all three phases (Figure 2 and the first panel of Figure 5). Its C_t reaches zero at round 3, accuracy peaks at round 4, and the decline begins at round 9. C_t therefore leads the accuracy peak by one round and the decline by six rounds, supporting its use as an early stopping signal. Section 4.3 verifies the same pattern across five additional methods.

Stopping Rule. We stop training once C_t remains below a threshold τ for two consecutive rounds, where $\tau = 0.1\, C_1$ is set relative to the first round. This point marks the end of Phase II, beyond which training enters Phase III. Training may optionally continue on external data, with samples selected using a directional variant of C_t that favors the model's current failure modes (Appendix J).

4 Experiments

4.1 Experimental Setup

Dataset. We follow the R-Diverse evaluation suite (Li et al., 2026) for direct comparability, comprising seven mathematical reasoning benchmarks (GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AMC, Minerva (Lewkowycz et al., 2022), Olympiad (He et al., 2024), AIME-2024, AIME-2025) and three general reasoning benchmarks (MMLU-Pro (Wang et al., 2024), SuperGPQA (Du et al., 2025), BBEH (Kazemi et al., 2025)). We report per-benchmark accuracy, seven-benchmark math average (Math AVG), and ten-benchmark overall average (Overall AVG).

Baselines. We compare ATRI against six self-evolution methods: *base*, *STaR* (Zelikman et al., 2022), *SPIN* (Chen et al., 2024), *AZR* (Zhao et al., 2025), *R-Zero* (Huang et al., 2025b), and *R-Diverse* (Li et al., 2026). All methods are run on the same base models and evaluated on the same suite for a controlled comparison.

Table 1: Main results: ATRI versus existing self-evolution methods on Qwen3-4B-Base and Qwen3-8B-Base across seven mathematical reasoning benchmarks and three general reasoning benchmarks. **Bold**: best per column. Underline: second best.

Method	Math AVG	Overall AVG	Mathematical Reasoning Benchmarks							General Reasoning		
			GSM8K	MATH	AMC	Minerva	Olymp.	AIME24	AIME25	MMLU-Pro	SuperGPQA	BBEH
Qwen3-4B-Base												
Base Model	42.58	36.34	87.86	67.96	45.59	38.10	41.16	11.03	6.35	37.17	20.84	7.33
STaR (Zelikman et al., 2022)	43.96	37.74	88.36	71.20	47.96	39.65	41.92	11.47	7.16	39.54	22.55	7.60
SPIN (Chen et al., 2024)	45.26	39.45	89.35	73.60	50.02	41.03	42.53	11.97	8.35	44.60	24.82	8.25
AZR (Zhao et al., 2025)	46.51	41.38	89.49	76.31	52.47	42.20	42.50	12.23	10.36	52.66	27.28	8.34
R-Zero (Huang et al., 2025b)	48.94	43.20	92.22	79.37	57.13	52.83	44.38	12.58	4.07	51.42	27.62	10.35
R-Diverse (Li et al., 2026)	52.56	46.20	92.28	78.65	60.11	**59.78**	**47.03**	**19.22**	10.88	55.35	28.37	10.29
ATRI	**53.55**	**47.39**	**93.24**	**81.20**	**61.42**	58.25	46.98	18.54	**15.20**	**56.48**	**30.50**	**12.09**
Qwen3-8B-Base												
Base Model	49.27	43.32	89.32	78.07	51.98	50.09	44.91	13.99	16.53	51.57	28.24	8.51
STaR (Zelikman et al., 2022)	49.96	44.05	89.66	78.62	53.29	51.11	45.28	14.40	17.36	53.18	28.83	8.82
SPIN (Chen et al., 2024)	51.44	45.57	90.88	79.18	55.44	54.05	46.35	16.06	18.10	56.40	29.90	9.32
AZR (Zhao et al., 2025)	52.68	47.23	91.80	76.66	58.04	57.86	47.58	18.34	18.45	60.51	32.13	10.98
R-Zero (Huang et al., 2025b)	54.65	48.30	93.85	82.11	61.76	60.68	48.77	16.42	18.98	58.20	31.36	10.83
R-Diverse (Li et al., 2026)	56.49	50.19	**94.58**	82.08	66.02	66.02	**49.99**	**24.67**	12.09	61.86	32.57	12.06
ATRI	**58.94**	**52.75**	94.47	**83.60**	**73.90**	**67.32**	49.42	22.85	**21.05**	**64.79**	**35.72**	**14.39**

Implementation details. We use Qwen3-4B-Base and Qwen3-8B-Base as base models. The proxy M_{t-1} used to compute C_t is Pythia-160M (Biderman et al., 2023), chosen for its small size and low computational cost while retaining sufficient capacity to fit the round-level data. It is fine-tuned on $\mathcal{D}_{t-1}$ for one epoch under the protocol of Section 3.1.2. The stopping threshold is $\tau = 0.1\, C_1$. Hyperparameters (learning rate, batch size, decoding settings, evaluation protocol, and seed counts) are listed in Appendix F. All models are trained 5 times, and we report mean performance.

4.2 Main comparison with existing self-evolution methods

We first compare the overall performance of ATRI with the self-evolution baselines. As shown in Table 1, ATRI achieves the best aggregate performance on both base models, and this advantage holds across both mathematical and general reasoning benchmarks, indicating that the improvement is not driven by any single task or model scale. We attribute this gain to the two complementary components of ATRI: within-round reweighting emphasizes more informative samples during training, while the cross-round C_t-based stopping rule halts evolution before it enters the degradation regime.

4.3 The three-phase lifecycle recurs across existing methods

Recurring lifecycle. We record the per-round accuracy and C_t trajectories of the six self-evolution baselines on Qwen3-4B-Base over 15 rounds. As shown in Figure 5, all methods exhibit a similar rise, plateau, and decline pattern. This trend is general rather than method-specific: a closed loop trains on its own outputs, so once $\mathcal{D}_t$ exhausts the model's reachable region, the round-level information gain saturates. Continuing past this point amounts to fine-tuning π_θ on samples it can already generate, which yields the same gradient as self-distillation onto the model's high-confidence region (Proposition 6) and contracts its solution distribution.

Round-level information gain C_t as an early signal. The same figure shows that C_t falls below the alarm threshold before the accuracy peak. Across all six methods, C_t falls below the alarm threshold τ one to four rounds before the peak, with larger lead times for methods with longer pre-peak plateaus. C_t directly reflects the redundancy between consecutive rounds of data and therefore changes earlier than accuracy. The decline in accuracy typically becomes visible only after further training on redundant data. The empirical results support the prediction in Section 3.3 that $C_t \to 0$ signals the end of Phase II, providing the basis for our stopping rule.

Direct evidence for Phase III contraction. Proposition 6 predicts that the solution distribution contracts during Phase III. We verify this on the Vanilla loop of Section 2. At five checkpoints, we sample eight solutions per problem on the same 500 held-out problems, and from these samples compute pass@1, pass@8, the entropy over distinct solution paths, and the number of distinct correct paths per problem. As shown in Table 2, all four metrics decline after the accuracy peak: pass@8 falls by 19.8 points, path entropy by 36.5%, and distinct correct paths by 46.6%. The loop increasingly concentrates on a smaller set of solutions while gaining little new information, consistent with Proposition 6. Appendix G extends this analysis with Llama-3.1-8B and with MBPP code generation. The lifecycle recurs in both settings, and the alarm again fires one round before the peak.

Table 2: Phase III contraction on the Vanilla loop (Qwen3-4B-Base, MATH). All metrics are computed from eight sampled solutions per problem on the same 500 held-out problems at each checkpoint.

Checkpoint	Round	Pass@1	Pass@8	Path entropy	Distinct correct paths
Before threshold	2	41.58	73.8	1.887	2.704
Threshold crossing	3	42.30	74.6	1.906	2.736
Accuracy peak	4	42.99	75.4	1.846	2.643
Phase III	8	38.28	68.6	1.537	2.087
Final round	15	28.93	55.6	1.173	1.412

Figure 4: C_t alarm vs. accuracy peak on the six methods of Figure 5. Orange: alarm round. Blue: lead to the accuracy peak.

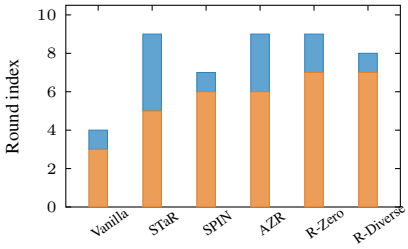
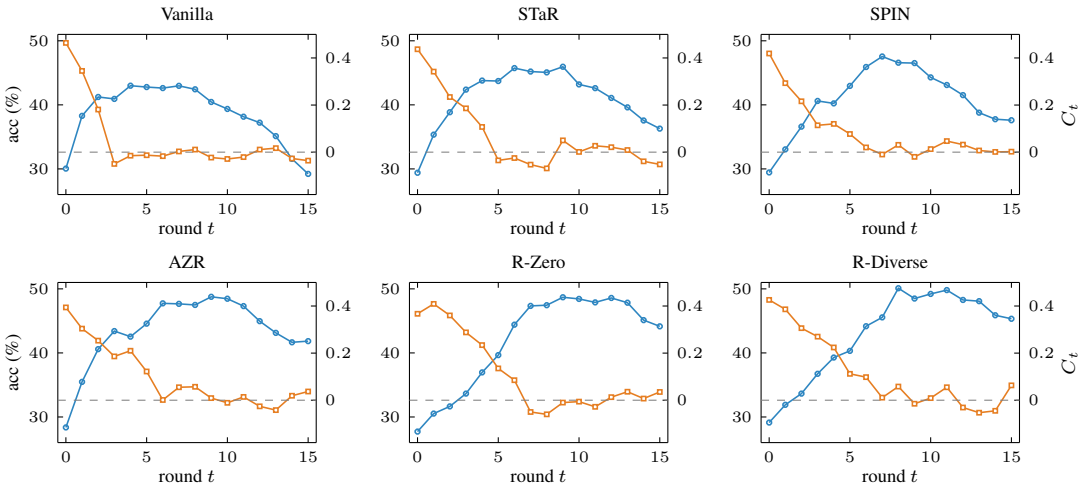


Figure 5: Three-phase lifecycle across self-evolution methods (Qwen3-4B-Base / MATH). C_t falls below the alarm threshold τ one to four rounds before the acc peak and then fluctuates around zero.



4.4 Direction-aware selection of external samples

When C_t approaches zero, the closed-loop data provide little additional learnable information, limiting further improvement. A common way to sustain learning is to introduce external data. We therefore investigate whether learnable information gain can be used to identify useful external samples. However, novelty alone is insufficient, since an external sample may be novel but already solvable by the model. ATRI therefore extends C_t into a directional learnable information gain C_t^d. At the stopping round, we divide the generated pairs into two groups based on the Solver's answers: correct and incorrect. We then train a separate proxy on each group. C_t^d of an external sample is its cost under the proxy of the correct part minus its cost under the proxy of the wrong part. A positive C_t^d means that the sample resembles what the model gets wrong. A sample is added when both C_t^d and the learnable information gain c_t of the sample are positive (Appendix J). Figure 6 compares this rule with *Random*, *Difficulty-only*, *R-Diverse* (Li et al., 2026), and c_t*-only* under the same budget. C_t^d-only beats all four at every budget. The combination C_t^d+c_t leads from 1500 samples on, with the widest margin at 3000. C_t^d finds the samples that resemble the model's failures. c_t then removes those the model has already absorbed.

4.5 Contribution of each component

ATRI's C_t control comprises two components: the stop rule and the within-round weighting and filtering. We vary them independently on Qwen3-4B-Base, yielding the four combinations in Table 4. Removing both components costs 3.37 Math AVG relative to the full method, and the two components contribute unequally to this gap. The stop rule accounts for the larger share: without it, training continues past the alarm round into Phase III, where accuracy reliably declines. The weighting and filtering contribute an additional 1.06 Math AVG on top of the stop rule under an identical stopping round, indicating that their benefit is independent of when training ends. Finally, adding the optional external phase on top of both components raises Math AVG by a further 2.12 (last row).

Figure 6: External-data selection comparison. Math AVG gain from the round-5 checkpoint vs. externally added samples.

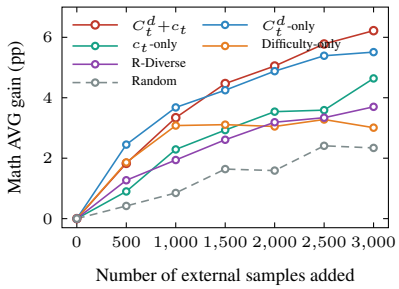


Table 3: Hyperparameter sensitivity on Qwen3-4B-Base. †: default.

Configuration	Math AVG	Overall AVG
Stopping threshold $\tau = \alpha C_1$		
$\alpha = 0.05$	53.18	47.02
$\alpha = 0.10^{\dagger}$	53.55	47.39
$\alpha = 0.20$	53.21	47.10
$\alpha = 0.30$	52.46	46.18
Proxy model size		
Pythia-70M	52.73	46.55
Pythia-160M†	53.55	47.39
Pythia-410M	53.62	47.45

4.6 Robustness and Computational Cost

Robustness. ATRI introduces two hyperparameters beyond the underlying training procedure: the stopping threshold $\tau = \alpha C_1$ and the proxy model size used to compute C_t. We sweep both around their defaults on Qwen3-4B-Base. As shown in Table 3, ATRI is robust within sensible ranges, with only the most aggressive setting losing accuracy. Raising α too far stops ATRI earlier than necessary, whereas lowering it delays the alarm and lets training drift into Phase III. Stopping slightly late costs less than stopping too early. The proxy size has an even smaller effect within the standard range, with the default at the elbow between underfitting and a 2.5-fold proxy cost. Appendix H provides additional stopping robustness checks.

Computational cost. The dominant cost of self-evolution is per-round generation and base-model fine-tuning. We measure the extra cost of ATRI's C_t layer. As shown in Table 5, C_t adds only a small fixed overhead from proxy training and per-sample scoring relative to the bare loop on Qwen3-8B-Base. This overhead is small because the Pythia-160M proxy cost does not scale with base-model size. As the base model grows, the relative overhead approaches the proxy-to-base parameter ratio, making the C_t control layer essentially free at current LLM scales. Measured GPU-hours show the overhead falling from 4.1% to 1.3% across three scales (Appendix K).

Table 4: Ablation of ATRI's C_t control on Qwen3-4B-Base.

Configuration	Math AVG	Δ_M	Overall AVG	Δ_O
Stop + filter (full ATRI)	53.55	—	47.39	—
Stop only	52.49	−1.06	46.31	−1.08
Filter only	51.27	−2.28	45.16	−2.23
Neither	50.18	−3.37	44.05	−3.34
With external phase	55.67	+2.12	49.60	+2.21

Table 5: Per-round computational cost on Qwen3-8B-Base, in PFLOPs.

Step	Vanilla	ATRI
Generation	16.2	16.2
Main fine-tuning	71.6	71.6
Proxy training	–	1.4
C_t computation	–	0.5
Total per round	**87.8**	**89.7**
Relative overhead	–	+2.2%

4.7 Case Study: Applying the Diagnostic to Existing Methods

We validate C_t as a monitor on the six Qwen3-4B-Base runs in Figure 5, tracking the signal passively during training. For each run, we record when it first falls below τ and when validation accuracy peaks. As shown in Figure 4, the alarm precedes the peak on all six baselines, with larger leads on longer pre-peak plateaus. This suggests that the signal detects dataset-level saturation in $\mathcal{D}_t$ before it is reflected in downstream accuracy. Because it depends only on $\mathcal{D}_t$ and a small reference proxy, the monitor applies unchanged to any self-evolution method. It is also actionable. Stopping once C_t stays below τ for two consecutive rounds raises average accuracy from 39.01 to 46.32, within 0.83 of the oracle test-best checkpoint, without held-out validation data (Appendix H).

5 Conclusion

Self-evolution often suffered from degeneration, yet existing remedies targeted isolated components. We introduced learnable information gain, a system-level diagnostic that quantified novel information in each round and, under an exact fit, decomposed into the KL divergence between consecutive data distributions plus their entropy change. Based on this diagnostic, we proposed ATRI, which reweighted samples within a round and halted training across rounds when information gain saturated. Experiments showed that ATRI mitigated degeneration and improved self-evolution.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting and language polishing (e.g., translation, LaTeX formatting), and to support literature retrieval and the discovery of related work. They also assisted in drafting and revising the proofs in the appendix. All AI-assisted content was reviewed by the authors, who take full responsibility for the final content of this paper.

REFERENCES

Emre Can Acikgoz, Cheng Qian, Jonas Hübotter, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Tool-R0: Self-evolving LLM agents for tool-learning from zero data. *arXiv preprint arXiv:2602.21320*, 2026.

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard G. Baraniuk. Self-consuming generative models go MAD. In *International Conference on Learning Representations*, 2024.

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L. Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models. In *International Conference on Learning Representations*, 2025.

Luke Bailey, Kaiyue Wen, Kefan Dong, Tatsunori Hashimoto, and Tengyu Ma. Scaling self-play with self-guidance. *arXiv preprint arXiv:2604.20209*, 2026.

Quentin Bertrand, Avishek Joey Bose, Alexandre Duplessis, Marco Jiralerspong, and Gauthier Gidel. On the stability of iterative retraining of generative models on their own data. In *International Conference on Learning Representations*, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Léonard Blier and Yann Ollivier. The description length of deep learning models. *Advances in Neural Information Processing Systems*, 31, 2018.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

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.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Elvis Dohmatob, Yunzhen Feng, Arjun Subramonian, and Julia Kempe. Strong model collapse. *arXiv preprint arXiv:2410.04840*, 2024a.

Elvis Dohmatob, Yunzhen Feng, Pu Yang, Francois Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws. *arXiv preprint arXiv:2402.07043*, 2024b.

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.

Marc Finzi, Shikai Qiu, Yiding Jiang, Pavel Izmailov, J. Zico Kolter, and Andrew Gordon Wilson. From entropy to epiplexity: Rethinking information for computationally bounded intelligence. *arXiv preprint arXiv:2601.03220*, 2026.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, et al. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. In *Conference on Language Modeling*, 2024.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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)*, pp. 3828–3850, 2024.

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. In *NeurIPS Datasets and Benchmarks*, 2021.

Audrey Huang, Adam Block, Dylan J. Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *International Conference on Learning Representations*, 2025a.

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*, 2025b.

Chengsong Huang, Haolin Liu, Tong Zheng, Runpeng Dai, Langlin Huang, Jinyuan Li, Zongxia Li, Zhepei Wei, Yu Meng, and Jiaxin Huang. G-Zero: Self-play for open-ended generation from zero data. *arXiv preprint arXiv:2605.09959*, 2026.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

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)*, pp. 26473–26501, 2025.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio du Pin Calmon. Inference-time reward hacking in large language models. *arXiv preprint arXiv:2506.19248*, 2025.

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.

Gengsheng Li, Jinghan He, Shijie Wang, Dan Zhang, Ruiqi Liu, Renrui Zhang, Zijun Yao, Junfeng Fang, Haiyun Guo, and Jinqiao Wang. R-diverse: Mitigating diversity illusion in self-play llm training. *arXiv preprint arXiv:2602.13103*, 2026.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *NAACL*, 2016.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7602–7635, 2024.

Jianzhe Lin. Self-improvement can self-regress: The rise-and-collapse failure mode of LLM self-training. *arXiv preprint arXiv:2606.21090*, 2026.

Sören Mindermann, Jan Brauner, Muhammed Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, 2022.

Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361, 2020.

Sophia Xiao Pu, Zhaotian Weng, Chengzhi Liu, Jayanth Srinivasa, Gaowen Liu, William Yang Wang, and Xin Eric Wang. Survive or collapse: The asymmetric roles of data gating and reward grounding in self-play RL. *arXiv preprint arXiv:2605.22217*, 2026.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Mohamed El Amine Seddik, Suei-Wen Chen, Soufiane Hayou, Pierre Youssef, and Merouane Debbah. How bad is training on synthetic data? a statistical analysis of language model collapse. *arXiv preprint arXiv:2404.05090*, 2024.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

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. *arXiv preprint arXiv:2402.03300*, 2024.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. *arXiv preprint arXiv:2412.02674*, 2024.

Yifan Sun, Yushan Liang, Zhen Zhang, Xin Liu, and Jiaye Teng. Theoretical modeling of large language model self-improvement training dynamics through solver-verifier gap. In *The Fourteenth International Conference on Learning Representations*, 2026.

Einar Urdshals, Edmund Lau, Jesse Hoogland, Stan van Wingerden, and Daniel Murfet. Compressibility measures complexity: Minimum description length meets singular learning theory. *arXiv preprint arXiv:2510.12077*, 2025.

Li Wang, Xiaodong Lu, Xiaohan Wang, Yikun Ban, Jiajun Chai, Wei Lin, Tianhao Peng, and Guojun Yin. When self-belief misleads: Active label acquisition for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2605.25864*, 2026.

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.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Ting Wu, Xuefeng Li, and Pengfei Liu. Progress or regress? self-improvement reversal in post-training. *arXiv preprint arXiv:2407.05013*, 2024.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, 2023.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-rewarding: Stable self-supervised RL for eliciting reasoning in large language models. In *International Conference on Learning Representations*, 2026.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, 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.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *SIGIR*, 2018.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. TTRL: Test-time reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.

A Related Work

Self-evolution and self-play for LLMs. Self-evolution covers methods in which an LLM improves from training signals it generates itself. Examples include self-training on filtered outputs (Zelikman et al., 2022; Huang et al., 2023; Gulcehre et al., 2023; Singh et al., 2024), self-rewarding (Yuan et al., 2024), self-play (Chen et al., 2024; Zhao et al., 2025; Huang et al., 2025b; Li et al., 2026), and reinforcement learning with majority-vote rewards (Zuo et al., 2025). Recent zero-data frameworks extend the Questioner–Solver loop to tool use (Acikgoz et al., 2026) and open-ended generation (Huang et al., 2026). Several of these works report gains that stall or reverse when training continues for many rounds (Huang et al., 2025b; Shafayat et al., 2025). Label-free reinforcement learning shows a similar training collapse (Zhang et al., 2026; Wang et al., 2026). Recent studies examine this collapse. Bailey et al. (2026) attribute the plateau of long self-play runs to a problem generator that hacks its reward. Pu et al. (2026) find that the gate deciding which generated tasks enter training matters more for stability than the reward design. Lin (2026) report a rise-then-collapse pattern in reinforcement learning on code and find that early stopping recovers part of the lost accuracy. These studies each focus on one component or one setting. A unified account of why the loop degrades is still missing. RAGEN (Wang et al., 2025) identified the "Echo Trap" failure mode and proposed reward variance as a diagnostic. Section 2 examines reward variance as a partial signal.

Model collapse and synthetic data degradation. The model collapse literature (Shumailov et al., 2024; Dohmatob et al., 2024b;a; Seddik et al., 2024; Alemohammad et al., 2024) shows that iterative training on self-generated data can lead to distributional collapse. Mixing in or accumulating real data can stabilize this process (Bertrand et al., 2024; Gerstgrasser et al., 2024). This work studies the self-evolution setting, where the loop generates both the questions and the answers. Generative Containment (Appendix B) gives an information-theoretic account of why a closed loop cannot add information on its own. The external phase of ATRI (Appendix J) selects which external samples to add.

Limits of self-improvement. Yue et al. (2025) showed that reinforcement learning with verifiable rewards mainly sharpens reasoning paths that the base model can already sample. Huang et al. (2025a) described self-improvement as sharpening the model toward its own high-likelihood outputs. Wu et al. (2024) reported a self-improvement reversal, in which pass@1 improves while output diversity and out-of-distribution generalization decline. Cui et al. (2025) linked the saturation of reinforcement learning for reasoning to the collapse of policy entropy. Song et al. (2024) introduced the generation-verification gap, and Sun et al. (2026) modeled self-improvement dynamics through the gap between solver and verifier. Generative Containment gives an information-theoretic account that is consistent with these findings.

Data selection with small reference models. Several data selection methods score training samples with a small reference model. RHO-LOSS (Mindermann et al., 2022) prioritizes points whose training loss is high but reducible, as judged by a model trained on holdout data. Ankner et al. (2025) prune pretraining data by the perplexity of a small reference model. DoReMi (Xie et al., 2023) uses small proxy models to set domain weights. Li et al. (2024) select instruction data by a model-based difficulty score. These methods score data against a fixed reference. The learnable information gain instead scores each round against the previous round of the same loop. The same score then weights samples and decides when to stop.

MDL and compression in deep learning. The MDL principle (Rissanen, 1978; Grünwald, 2007) has been applied to deep networks through prequential coding (Blier & Ollivier, 2018) and connected to singular learning theory (Urdshals et al., 2025). Finzi et al. (2026) define epiplexity as the information that a computationally bounded observer can learn from data and use it to guide data selection. This work applies the MDL view to the dynamics of self-evolution and derives a diagnostic from it.

Self-distillation theory. Mobahi et al. (2020) proved that repeated self-distillation progressively limits the expressiveness of the solution. Proposition 6 relates Phase III to this result. Further training after the learnable information gain is exhausted behaves like self-distillation.

B Information-theoretic Boundary of Self-Evolution

The inconsistency observed in Section 2 suggests that partial signals miss the direction of the loop because the loop itself is subject to a constraint that those signals do not measure. We formalize this constraint here.

Let P_t denote the model parameters at round t, treated as a random variable, and let X denote the target capability the model is trying to acquire. In each round, the model generates training data $Q_t = G(P_t)$ through a generation function G. It then updates to $P_{t+1} = T(P_t, Q_t)$ through a training function T. The update starts from P_t, so P_{t+1} depends on both P_t and Q_t. Assume that G and T use no signal about X beyond P_t. Their internal randomness, such as sampling noise, is independent of X. This gives the Markov chain

$$X \;\longrightarrow\; P_t \;\longrightarrow\; (P_t, Q_t) \;\longrightarrow\; P_{t+1}. \tag{16}$$

The data Q_t are a function of P_t and independent noise. By the data processing inequality (DPI), processing a variable cannot increase its mutual information with a third variable, so

$$I(Q_t;\, X) \;\le\; I(P_t;\, X). \tag{17}$$

Since $P_{t+1} = T(P_t, Q_t)$, applying the DPI again yields

$$I(P_{t+1};\, X) \;\le\; I(P_t, Q_t;\, X) \;=\; I(P_t;\, X). \tag{18}$$

The equality holds because Q_t carries no information about X beyond P_t. Together, (17) and (18) give

$$\max\big\{I(Q_t;\, X),\; I(P_{t+1};\, X)\big\} \;\le\; I(P_t;\, X), \tag{19}$$

where $I(\cdot\,;\cdot)$ denotes mutual information.

The inequality says that the loop produces no new information about X from within. Whatever information each round uses is inherited from the previous round. Generation and training can only preserve or reduce it. We call this structural constraint *Generative Containment*. It does not depend on the choice of G or T and follows from the Markov structure of the chain (16) alone. It therefore holds for any self-evolution algorithm that meets the assumption above. Signals from outside the loop, such as external data or ground-truth verification, break the assumption and can add information about X.

Generative Containment describes one boundary of the information budget. The budget cannot grow on its own. It does not describe how the budget is spent, when it runs out, or what state the model enters once it does. The constructions in Section 3 fill in this missing dynamics by tracking how much of the budget is still available for learning at each round.

C Within-round Decomposition of C_t

This appendix derives the within-round decomposition of C_t along the Question–Solver generation order. It splits C_t into a question term and a solution term. It also restates the per-segment gains $c_t^Q(q)$ and $c_t^S(s \mid q)$ of Sections 3.1 and 3.2.

In the common two-step Question–Solver setting, the model first generates a question q and then a solution s conditioned on q, producing $(q, s) \in \mathcal{D}_t$. Since ℓ is defined token by token, it extends to any contiguous segment. We write $\ell(q; P_{t-1})$ for the average NLL over the question tokens and $\ell(s \mid q; P_{t-1})$ for the average NLL over the solution tokens with q as prefix. Per-token averages are not additive, so the decomposition carries length weights. With $\lambda_e = |q|/(|q| + |s|)$, multiplying the chain rule $\log P_{t-1}(q, s) = \log P_{t-1}(q) + \log P_{t-1}(s \mid q)$ by $-1/(|q| + |s|)$ gives

$$\ell(q, s; P_{t-1}) = \lambda_e\, \ell(q; P_{t-1}) + (1 - \lambda_e)\, \ell(s \mid q; P_{t-1}). \tag{20}$$

Substituting (20) into the definition of C_t (Eq. (15)) and averaging over $\mathcal{D}_t$ and $\mathcal{D}_{t-1}$ separately gives

$$C_t = \big[\ell_q^\lambda(\mathcal{D}_t) - \ell_q^\lambda(\mathcal{D}_{t-1})\big] + \big[\ell_{s|q}^\lambda(\mathcal{D}_t) - \ell_{s|q}^\lambda(\mathcal{D}_{t-1})\big], \tag{21}$$

where $\ell_q^\lambda(\mathcal{D}) = \mathbb{E}_{(q,s)\in\mathcal{D}}[\lambda_e\,\ell(q;P_{t-1})]$ and $\ell_{s|q}^\lambda(\mathcal{D}) = \mathbb{E}_{(q,s)\in\mathcal{D}}[(1-\lambda_e)\,\ell(s\mid q;P_{t-1})]$ are the length-weighted dataset averages. The decomposition is exact. The first term measures how novel the questions of round t are relative to round $t-1$. The second measures how novel the solutions are, conditioned on the corresponding questions.

At the sample level, we define the per-segment gains $c_t^Q(q) = \ell(q;P_{t-1}) - \ell_q(\mathcal{D}_{t-1})$ and $c_t^S(s\mid q) = \ell(s\mid q;P_{t-1}) - \ell_{s|q}(\mathcal{D}_{t-1})$, where ℓ_q and $\ell_{s|q}$ are the unweighted dataset averages of the segment NLLs. Their positive parts are the weights in Equations (11) and (14).

D Proxy Model Capacity Selection

This appendix gives the rationale behind our choice of the proxy language model M_{t-1} used to instantiate P_{t-1} in Section 3.1.2, and discusses why the result reported in Table 3 is largely insensitive within a wide range of proxy capacities.

Design requirement. The proxy serves a single purpose. It distinguishes samples *inside* $\mathcal{D}_{t-1}$ from samples *outside*. After fitting M_{t-1} on $\mathcal{D}_{t-1}$, the encoding cost $\ell_{M_{t-1}}(\cdot)$ should be (i) low and stable on $\mathcal{D}_{t-1}$, providing the baseline $\ell(\mathcal{D}_{t-1};P_{t-1})$, and (ii) sensitive to deviations, so that samples from a different distribution receive a measurably higher cost. A proxy that fails (i) gives a noisy baseline. A proxy that fails (ii) cannot detect novelty.

Capacity trade-off. An over-parameterised proxy memorises $\mathcal{D}_{t-1}$ together with substantial neighbourhoods of it, so $\ell_{M_{t-1}}$ assigns low cost not just to $\mathcal{D}_{t-1}$ but to a much broader region. Criterion (ii) degrades, and C_t shrinks toward zero on every input regardless of round. An under-capacity proxy underfits $\mathcal{D}_{t-1}$. The baseline $\ell(\mathcal{D}_{t-1};P_{t-1})$ becomes a noisy estimate, criterion (i) degrades, and C_t inherits this noise as round-to-round jitter that obscures the lifecycle.

Why the Pythia family. We use Pythia (Biderman et al., 2023) for three reasons. (i) The suite spans 70M to 12B parameters trained on a single fixed corpus (the Pile) under a single recipe, so capacity can be varied as a controlled axis without confounders from training data or recipe. (ii) The family is small enough at the lower end (70M / 160M / 410M) that one-epoch fine-tuning on $\mathcal{D}_{t-1}$ adds negligible compute relative to the main model fine-tune (Table 5). (iii) The vocabulary and tokenizer remain fixed across sizes, so that encoding-cost values from different proxy sizes are directly comparable.

Empirical sweet spot. Table 3 reports a sweep over Pythia-{70M, 160M, 410M}. Math AVG moves from 52.73 (70M) to 53.55 (160M) to 53.62 (410M). The 70M proxy underfits and adds noise. The 410M proxy is marginally better than 160M but raises proxy compute about 2.5-fold. The 160M default sits at the elbow. Above 1B the proxy-fitting cost begins to compete with main-model fine-tuning, and the over-parameterised regime would erode criterion (ii).

One-epoch fine-tuning. The proxy M_{t-1} is fine-tuned on $\mathcal{D}_{t-1}$ for exactly one epoch from a fresh checkpoint at each round. Multi-epoch training drives $\ell_{M_{t-1}}$ closer to the global minimum on $\mathcal{D}_{t-1}$ but also expands the low-cost region around it, again degrading criterion (ii). One epoch is the minimum sufficient pass for the proxy to acquire $\mathcal{D}_{t-1}$'s gross statistics without memorising specific samples.

Decoupling from the main model. The proxy is a separate model that restarts from the pretrained Pythia checkpoint at every round. It never sees the main model's parameters. As a consequence, C_t depends only on $\mathcal{D}_{t-1}$, $\mathcal{D}_t$, and the chosen proxy capacity. It does not depend on the main model's parameter scale, training algorithm, sampling temperature, or random seed. This decoupling is what makes C_t a method-agnostic monitor in the sense of Section 4.7.

E Extensions of the Calibration Loss

Sections 3.1 and 3.2 train the Questioner and the Solver as two models with GRPO. This appendix gives two extensions. The first is the shared-parameter case, where one model plays both roles. The

second covers supervised fine-tuning and DPO. The principle is the same in both. The positive part of the gain rescales the per-sample term of the underlying objective.

E.1 Shared-parameter case

When one model π_θ generates both the question and the solution, θ and ϕ coincide. The losses in Equations (11) and (14) then act on the same parameters,

$$\mathcal{L}(\theta) = \mathcal{L}^Q(\theta) + \mathcal{L}^S(\theta). \tag{22}$$

The gradients of both sides enter θ together. The weights w_t^Q and w_t^S stay the same.

E.2 Supervised fine-tuning

Under SFT, each training pair enters the negative log-likelihood with weight one. The calibrated form replaces this weight with w_t^S,

$$\mathcal{L}_{\mathrm{cal}}^{\mathrm{SFT}}(\phi) = -\frac{1}{|\mathcal{D}_t|} \sum_{(q,s)\in\mathcal{D}_t} w_t^S(s \mid q) \log S_\phi(s \mid q). \tag{23}$$

The Questioner side weights $\log Q_\theta(q \mid p_0)$ by $w_t^Q(q)$ in the same way. A sample whose gain is not positive receives zero weight.

E.3 DPO

In the self-play DPO setting (Chen et al., 2024), each training instance is a preference pair (e^+, e^-), where $e^+ = (q, s^+)$ is preferred over $e^- = (q, s^-)$. The standard DPO loss is

$$\mathcal{L}_{\mathrm{DPO}}(\phi) = -\mathbb{E}_{(e^+,e^-)}\left[\log\sigma\Big(\beta\log\tfrac{S_\phi(s^+|q)}{S_{\mathrm{ref}}(s^+|q)} - \beta\log\tfrac{S_\phi(s^-|q)}{S_{\mathrm{ref}}(s^-|q)}\Big)\right]. \tag{24}$$

The calibrated form weights each pair by the gain of its preferred solution,

$$\mathcal{L}_{\mathrm{cal}}^{\mathrm{DPO}}(\phi) = -\mathbb{E}_{(e^+,e^-)}\left[w_t^S(s^+ \mid q)\cdot\log\sigma\Big(\beta\log\tfrac{S_\phi(s^+|q)}{S_{\mathrm{ref}}(s^+|q)} - \beta\log\tfrac{S_\phi(s^-|q)}{S_{\mathrm{ref}}(s^-|q)}\Big)\right]. \tag{25}$$

When $w_t^S(s^+ \mid q)$ is small, s^+ is already covered by $\mathcal{D}_{t-1}$ and the pair contributes little gradient. When it is large, the pair drives the update. The negative branch e^- needs no separate weight because it only provides contrast for e^+.

When the Questioner is trained with a preference objective, the same scheme uses $w_t^Q(q^+)$,

$$\mathcal{L}_{\mathrm{cal}}^{\mathrm{DPO},Q}(\theta) = -\mathbb{E}_{(q^+,q^-)}\left[w_t^Q(q^+)\cdot\log\sigma\Big(\beta\log\tfrac{Q_\theta(q^+|p_0)}{Q_{\mathrm{ref}}(q^+|p_0)} - \beta\log\tfrac{Q_\theta(q^-|p_0)}{Q_{\mathrm{ref}}(q^-|p_0)}\Big)\right]. \tag{26}$$

E.4 General principle

For any training objective of the form $\mathcal{L} = \mathbb{E}_e\big[w(e)\cdot\ell_{\mathrm{base}}(e)\big]$, the calibrated version replaces $w(e)$ with $w_t(e)\cdot w(e)$, where w_t is the positive part of the gain. The base weight $w(e)$ is 1 for SFT, the advantage for GRPO, and the implicit log-sigmoid gradient for DPO. The two-segment decomposition of Appendix C applies whenever ℓ_{base} admits a corresponding decomposition.

F Implementation Details

This appendix collects the hyperparameters and protocol details omitted from Section 4. Default values are reported. Any deviation in a specific experiment is noted at the relevant location in the main text.

Main model fine-tuning. The Questioner and the Solver are trained with the GRPO objectives of Sections 3.1 and 3.2. For both Qwen3-4B-Base and Qwen3-8B-Base, the optimizer is AdamW (learning rate 1×10^{-5}, β_1=0.9, β_2=0.95, weight decay 0.1), a constant schedule with 50-step linear warmup, batch size 128 (gradient accumulation as needed), and bfloat16 mixed precision. Each round trains for one epoch over the round's $\mathcal{D}_t$ (or external pool E in the external phase). Sequence length is capped at 2048 tokens. Gradient clipping is set to 1.0.

Generation. Sampling uses temperature 0.7, top-p 0.95, and a maximum of 2048 generated tokens. For each question the Solver draws 8 answers per round, matching all baselines. The verifier-positive answers form $\mathcal{D}_t^+$ and the verifier-negative answers form $\mathcal{D}_t^-$.

Verifier. Answers are extracted and normalised before matching (numerical equivalence, fraction simplification, LaTeX normalisation). In the ATRI loop, each answer is matched against the majority-vote pseudo-label $\tilde{y}_i$ of Section 3.2. The Vanilla loop of Section 2 matches against the ground-truth answer. No learned reward model is used. On general benchmarks no online verifier is needed because evaluation is offline.

Proxy fine-tuning. The proxy M_{t-1} (Pythia-160M by default) is fine-tuned on $\mathcal{D}_{t-1}$ for one epoch from a fresh Pythia checkpoint at the start of each round, with AdamW (lr 5×10^{-5}), batch size 64, sequence length 1024, and bfloat16. The directional proxies M_t^+ and M_t^- use the same setup applied to $\mathcal{D}_t^+$ and $\mathcal{D}_t^-$. Proxy training adds 1.4 PFLOPs per round on Qwen3-8B-Base configurations (Table 5).

Stop threshold. The alarm threshold is $\tau = \alpha\, C_1$ with default $\alpha = 0.1$, where C_1 is the first-round learnable information gain. The closed loop ends when $C_t < \tau$ for two consecutive rounds. Table 3 sweeps $\alpha \in \{0.05, 0.10, 0.20, 0.30\}$.

External pool. The external pool E for the external phase is drawn from open-source math corpora (NuminaMath, MetaMathQA, OpenMathInstruct) with deduplication against the closed-loop questions. For each external candidate only the question is retained. The solution is generated and verified online by the current main model, just as in the closed-loop phase. The dual-gate filter $\{c_t(e) > 0,\ C_t^d(e) > 0\}$ is applied before the sample enters training.

Baselines. We use each baseline's official implementation when available (STaR, SPIN, AZR, R-Zero, R-Diverse) and re-run it on the same base models, generation budget, verifier, and evaluation protocol used for ATRI. *Vanilla* is implemented in-house as the bare four-step loop without selection or reward shaping. The external pool is used only by ATRI's external phase and the selection study of Section 4.4. No baseline trains on it.

Evaluation. Per-benchmark accuracy is computed with greedy decoding (temperature 0) and standard zero-shot prompts. Math AVG and Overall AVG are unweighted means over the seven and ten benchmarks respectively. All numbers in Table 1 are means over 5 training runs. Standard deviations are within ± 0.4 on Math AVG.

G Generality Across Model, Task, and Proxy Families

The main experiments use Qwen models on mathematical reasoning with a Pythia proxy. This appendix repeats the lifecycle analysis in two new settings. Llama-3.1-8B on MATH changes the base-model family. Qwen3-4B on MBPP changes the task to code generation. The Pythia-160M proxy and the threshold $\tau = 0.1\, C_1$ stay fixed.

Table 6: Lifecycle and alarm behaviour across base models and tasks. Accuracy is held-out performance at the listed round.

Setting	Task	Base model	Peak round	Alarm round	Peak acc.	Round-15 acc.
Reference	MATH	Qwen3-4B	4	3	42.99	29.22
New model family	MATH	Llama-3.1-8B	5	4	45.16	37.84
New task type	MBPP	Qwen3-4B	6	5	59.10	51.70

Both new trajectories rise and then decline. The C_t alarm fires one round before the peak in every setting. The lifecycle and the alarm timing therefore also appear beyond Qwen models and mathematical reasoning.

Proxy family. OPT-125M replaces Pythia-160M on the same Qwen3-4B/MATH trajectory. The main-model checkpoints and the retained datasets stay unchanged. The alarm fires at the same round. The sample weights of the two proxies have a Spearman correlation of 0.913. Running full ATRI with the OPT proxy changes the final scores by at most 0.14 points.

H Robustness of the Stopping Decision

This appendix compares the C_t stopping decision with alternative stopping rules. It also tests how the decision behaves under smaller per-round datasets, surface-level variation, task shifts, and round-free training. Unless stated otherwise, all experiments use the Qwen3-4B/MATH full-ATRI trajectory.

Stopping quality. Four stopping rules are compared on the six 15-round baseline runs monitored in Section 4.7, keeping every training procedure unchanged. Round 15 takes the final checkpoint. Validation stopping returns the best checkpoint once held-out accuracy fails to improve for two rounds. Oracle checkpointing picks the test-best checkpoint and gives an upper bound. Passive C_t stopping returns the checkpoint at which C_t has stayed below τ for two consecutive rounds. It uses the monitored C_t alone and needs no validation data. Table 7 reports the accuracy of each returned checkpoint.

Table 7: MATH accuracy of the checkpoint returned by four stopping rules on the six baseline runs of Section 4.7. The selected round is shown in parentheses.

Method	Round 15	Validation stopping	Oracle checkpointing	Passive C_t stopping
Vanilla	28.93 (15)	42.99 (4)	42.99 (4)	42.99 (4)
STaR	36.37 (15)	45.18 (8)	45.43 (9)	44.87 (6)
SPIN	37.41 (15)	47.55 (7)	47.55 (7)	46.61 (6)
AZR	41.58 (15)	48.11 (8)	48.27 (9)	47.22 (7)
R-Zero	44.32 (15)	48.62 (10)	48.74 (9)	47.86 (8)
R-Diverse	45.47 (15)	49.58 (9)	49.91 (8)	48.36 (7)
Average	39.01 (15.0)	47.01 (7.7)	47.15 (7.7)	46.32 (6.3)

Finite-sample stability. Small per-round datasets make C_t noisier. To measure the effect on the stopping decision, each round is bootstrap-resampled at four sizes. At each size, 1,000 trials recompute C_t and apply the same two-consecutive-round stopping rule. Table 8 reports how often the trials reproduce the full-data stopping round. Even at 25% of the original size, 77.6% of the trials reproduce it. Every trial stays within two rounds of the full-data decision.

Table 8: Bootstrap stability of the stopping round at four per-round data sizes. Each row summarises 1,000 trials.

Samples per round	Trials matching full-data round	Maximum shift
100%	94.2%	1 round
75%	91.3%	1 round
50%	85.8%	1 round
25%	77.6%	2 rounds

Surface-level controls. C_t could in principle react to sample length or formatting rather than to new content. Three controls test this. Length matching equalises the length distributions of adjacent rounds and is repeated 100 times. Format standardisation normalises prompt templates, whitespace, and equivalent LaTeX forms. The third setting combines both. Table 9 shows that every control keeps the original alarm and stopping rounds. The combined control stays at a correlation of 0.958 with the original trajectory and selects the same stopping round in 93.1% of trials. The remaining trials differ by one round.

Task shifts. A proxy fitted on one task may stop tracking the gain after the loop switches to another task. A run that switches from MATH to MBPP mid-loop tests this. One setting keeps the proxy fitted on the final MATH round. The other refits the proxy on the first MBPP round. Table 10 reports

Table 9: Stopping behaviour under surface-level controls. Correlation is Spearman against the original C_t trajectory. Ranges are standard deviations over the 100 length-matching repetitions.

Setting	Correlation	Alarm shift	Stopping shift	Same stopping round
Original	1.000	0	0	100%
Length matched	0.982 ± 0.007	0	0	95.6%
Format standardised	0.969	0	0	100%
Both controls	0.958 ± 0.010	0	0	93.1%

both settings next to the single-task runs of Table 6. With the old-task proxy the alarm fires two rounds after the performance peak. Refitting moves the alarm back to one round before the peak. This matches the single-task runs. After a task switch the proxy should therefore be refitted on the first round of the new task.

Table 10: Alarm timing under task shifts. The last column counts the rounds by which the alarm precedes the performance peak.

Setting	Proxy data	Alarm round	Peak round	Rounds before peak
MATH	Previous MATH round	3	4	1
MBPP	Previous MBPP round	5	6	1
MATH $\rightarrow$ MBPP	Final MATH round	8	6	-2
MATH $\rightarrow$ MBPP	First MBPP round	5	6	1

Round-free training. Streaming and online settings have no round boundaries. The diagnostic only needs a data window, so rounds can be replaced by fixed token windows. On the same trajectory, windows of half and one times the average round size stop at the same checkpoint as the round-based rule. A window of two rounds stops one checkpoint later. A window on the order of one round of data is a reasonable default.

I PROOFS

This appendix collects the formal statements supporting the claims of Section 3. We use the notation of Section 3. $\mathcal{D}_t$ is the round-t training set, and P_{t-1} is the reference distribution fitted on $\mathcal{D}_{t-1}$. The encoding cost $\ell(x; P) = -\frac{1}{|x|}\sum_i \log P(x_i \mid x_{<i})$ is the token-normalized negative log-likelihood of Section 3.1.2. The main text writes $\ell(x)$ and $\bar{\ell}(\mathcal{D})$ for $\ell(x; P_{t-1})$ and $\ell(\mathcal{D}; P_{t-1})$, with P_{t-1} given by the proxy M_{t-1}.

I.1 BOUNDARY CONDITION

Proposition 1 (Boundary condition)**.** *If $\mathcal{D}_t$ and $\mathcal{D}_{t-1}$ have the same empirical distribution, then $C_t = 0$.*

Proof. By definition $C_t = \ell(\mathcal{D}_t; P_{t-1}) - \ell(\mathcal{D}_{t-1}; P_{t-1})$, where $\ell(\mathcal{D}; P) = \mathbb{E}_{x\sim\mathcal{D}}[\ell(x; P)]$. The two averages are taken over the same empirical distribution, so they coincide and $C_t = 0$. □

I.2 INFORMATION-THEORETIC IDENTITY

Proposition 2 (General identity)**.** *Let p_t and p_{t-1} denote the distributions underlying $\mathcal{D}_t$ and $\mathcal{D}_{t-1}$. For a distribution p over sequences, let $H(p) = \mathbb{E}_{x\sim p}[\ell(x; p)]$ and $\mathrm{KL}(p \,\|\, q) = \mathbb{E}_{x\sim p}\big[\frac{1}{|x|}\log\frac{p(x)}{q(x)}\big]$ denote the per-token entropy and KL divergence. For any reference distribution P_{t-1}, replacing the empirical averages in Equation* (15) *with expectations gives*

$$C_t = \big[\mathrm{KL}(p_t \,\|\, P_{t-1}) - \mathrm{KL}(p_{t-1} \,\|\, P_{t-1})\big] + \big[H(p_t) - H(p_{t-1})\big].$$

Proof. For any pair of distributions p, q, the expected encoding cost decomposes as

$$\mathbb{E}_{x\sim p}\big[\ell(x; q)\big] = \mathbb{E}_{x\sim p}\Big[-\tfrac{1}{|x|}\log p(x)\Big] + \mathbb{E}_{x\sim p}\Big[\tfrac{1}{|x|}\log\tfrac{p(x)}{q(x)}\Big] = H(p) + \mathrm{KL}(p \,\|\, q).$$

Applying this with $(p, q) = (p_t, P_{t-1})$ to the first term of Equation (15), with $(p, q) = (p_{t-1}, P_{t-1})$ to the second term, and subtracting, yields the claim. □

Corollary 1 (Exact-fit limit)**.** *If the reference matches the previous-round distribution, $P_{t-1} = p_{t-1}$, then $C_t = \mathrm{KL}(p_t \,\|\, p_{t-1}) + \big[H(p_t) - H(p_{t-1})\big]$.*

Proof. Immediate from Proposition 2, since $\mathrm{KL}(p_{t-1} \,\|\, p_{t-1}) = 0$. □

Remark 1 (Robustness to an under-fitted reference)**.** *Proposition 2 holds for any reference P_{t-1}. This explains why the small one-epoch proxy of Section 3.1.2 is sufficient. Two points are worth noting.*

(i) The zero point is exact. *If $p_t = p_{t-1}$, the two KL terms are equal and the entropy difference is zero. So $C_t = 0$ for any P_{t-1}. The reason is the baseline subtraction in Equation* (15)*. The proxy's fitting error appears in both terms and cancels. The stopping rule monitors exactly this point. This is the population form of Proposition 1.*

(ii) Away from zero, only the residual difference matters. *Let $r(x) = \frac{1}{|x|} \log \frac{p_{t-1}(x)}{P_{t-1}(x)}$ be the proxy's fitting residual on x. The first bracket in Proposition 2 equals $\mathrm{KL}(p_t \,\|\, p_{t-1}) + \mathbb{E}_{p_t}[r] - \mathbb{E}_{p_{t-1}}[r]$. The gap to Corollary 1 is the residual difference between the two rounds. This difference is small when the residual is low and stable on the region shared by both rounds, which is criterion (i) of Appendix D. Sensitivity to novel samples is criterion (ii). The capacity trade-off of Appendix D therefore serves the identity. The proxy should have a stable residual where the rounds overlap, and a large cost outside.*

All quantities above are averaged per token. When the two rounds have the same length distribution, $\mathrm{KL}(p_t \,\|\, p_{t-1})$ is nonnegative. The entropy change can still be negative. A contracting loop lowers $H(p_t)$, and this alone can make C_t negative even under exact fit. Length shifts, the residual difference in (ii), and finite samples can also push the empirical C_t slightly below zero. These effects match the small negative values in Figure 5.

I.3 Why the gain decreases in a closed loop

Proposition 3 (Decreasing gain in a contracting loop)**.** *Assume the exact-fit setting of Corollary 1 at every round. Suppose*

$$\mathrm{KL}(p_{t+1} \,\|\, p_t) \le \mathrm{KL}(p_t \,\|\, p_{t-1}) \qquad \text{and} \qquad H(p_{t+1}) - H(p_t) \le H(p_t) - H(p_{t-1}). \tag{27}$$

Then $C_{t+1} \le C_t$. If moreover $\mathrm{KL}(p_{t+1} \,\|\, p_t) \to 0$ and $H(p_{t+1}) - H(p_t) \to 0$, then $C_t \to 0$.

Proof. By Corollary 1,

$$C_{t+1} - C_t = \big[\mathrm{KL}(p_{t+1} \,\|\, p_t) - \mathrm{KL}(p_t \,\|\, p_{t-1})\big] + \big[(H(p_{t+1}) - H(p_t)) - (H(p_t) - H(p_{t-1}))\big].$$

Both brackets are nonpositive under condition (27). The limit statement follows by reading $C_{t+1} = \mathrm{KL}(p_{t+1} \,\|\, p_t) + [H(p_{t+1}) - H(p_t)]$ directly. □

Remark 2 (When the conditions hold)**.** *Condition* (27) *says the loop contracts. Successive rounds move less, and the entropy change does not grow. A closed loop has this property because the model trains only on its own outputs. Repeated training of this form shrinks the effective hypothesis space (Mobahi et al., 2020). Proposition 6 shows the same mechanism in our setting. The condition fails once external data enters. New data enlarges the shift $\mathrm{KL}(p_{t+1} \,\|\, p_t)$, so the gain can rise again. This is why the external phase of Appendix J restores the gain. Finally, the proposition assumes an exactly fitted reference. With the underfit proxy of Section 3.1.2, the comparison holds up to the residual differences of Remark 1.*

I.4 Sample average equals the dataset gain

Proposition 4 (Sample-average identity)**.** $\frac{1}{|\mathcal{D}_t|} \sum_{e \in \mathcal{D}_t} c_t(e) = C_t$.

Proof. By definition,

$$c_t(e) = \ell(e; P_{t-1}) - \ell(\mathcal{D}_{t-1}; P_{t-1}).$$

Averaging over $e \in \mathcal{D}_t$,

$$\frac{1}{|\mathcal{D}_t|}\sum_{e\in\mathcal{D}_t} c_t(e) = \underbrace{\frac{1}{|\mathcal{D}_t|}\sum_{e\in\mathcal{D}_t} \ell(e; P_{t-1})}_{=\ell(\mathcal{D}_t;P_{t-1})} - \ell(\mathcal{D}_{t-1}; P_{t-1}) = C_t.$$

The proposition justifies using C_t as the round-level summary and $c_t(e)$ as its sample-level decomposition (Section 3.1.2). □

I.5 Question–Solver decomposition

Proposition 5 (Two-step decomposition)**.** *For samples $(q, s) \in \mathcal{D}_t$ generated by a two-step Question–Solver process,*

$$C_t = \underbrace{\left[\ell_q^\lambda(\mathcal{D}_t) - \ell_q^\lambda(\mathcal{D}_{t-1})\right]}_{C_t^Q} + \underbrace{\left[\ell_{s|q}^\lambda(\mathcal{D}_t) - \ell_{s|q}^\lambda(\mathcal{D}_{t-1})\right]}_{C_t^S},$$

where ℓ_q^λ and $\ell_{s|q}^\lambda$ are the length-weighted dataset averages defined in Appendix C.

Proof. The chain rule of conditional probability gives

$$\log P_{t-1}(q, s) = \log P_{t-1}(q) + \log P_{t-1}(s \mid q).$$

Multiplying by $-1/(|q| + |s|)$ and using the definition of ℓ yields Equation (20). Averaging over $\mathcal{D}_t$ and $\mathcal{D}_{t-1}$ and substituting into the definition of C_t yields the claim. □

I.6 Connection to self-distillation in Phase III

Proposition 6 (Phase III as self-distillation)**.** *Suppose two conditions hold at every round $t \geq T$. (i) $C_t \leq 0$. (ii) The main model π_θ has fitted $\mathcal{D}_{t-1}$ well enough to place its high-probability mass on* $\mathrm{supp}(\mathcal{D}_{t-1})$*, that is, it has low cross-entropy on $\mathcal{D}_{t-1}$. Then for every $t \geq T$, the SFT update of π_θ on $\mathcal{D}_t$ is approximately a self-distillation step of π_θ in the sense of Mobahi et al. (2020).*

Proof. We show in three steps that the SFT loss on $\mathcal{D}_t$ becomes a self-referential objective on π_θ, and then invoke Mobahi et al. (2020) for the self-distillation interpretation.

Step 1: $\mathcal{D}_t$ is contained in the proxy's view of $\mathrm{supp}(\mathcal{D}_{t-1})$. By assumption (i) and the definition of C_t in (15),

$$\ell(\mathcal{D}_t; P_{t-1}) \;\leq\; \ell(\mathcal{D}_{t-1}; P_{t-1}).$$

The reference P_{t-1} is fitted on $\mathcal{D}_{t-1}$, so its encoding cost attains a low plateau on samples drawn from $\mathcal{D}_{t-1}$ and grows on samples outside it. The inequality therefore implies that $\mathcal{D}_t$ is supported, on average, within the level set

$$\mathcal{S}_{t-1} \;\triangleq\; \{x : \ell(x; P_{t-1}) \leq \ell(\mathcal{D}_{t-1}; P_{t-1})\},$$

which is the proxy's characterization of $\mathrm{supp}(\mathcal{D}_{t-1})$. In the limiting case $C_t = 0$, the inequality holds with equality.

Step 2: $\mathcal{D}_t$ lies in π_θ's high-probability region. By assumption (ii), π_θ assigns its high-probability mass to $\mathrm{supp}(\mathcal{D}_{t-1})$. By Step 1, $\mathcal{D}_t$ lies mostly within this region. Most $x \in \mathcal{D}_t$ therefore satisfy $\pi_\theta(x) \geq \rho$ for a threshold ρ set by how well round $t-1$ was fitted. The training set is thus drawn mainly from the high-probability region of the same model that is being trained.

Step 3: the SFT loss reduces to a self-referential objective. The SFT loss is

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; -\mathbb{E}_{x\sim\mathcal{D}_t}[\log \pi_\theta(x)].$$

Since $\mathcal{D}_t$ is sampled from a distribution concentrated on π_θ's own high-probability region (Step 2), the empirical expectation approximates the population expectation taken under π_θ restricted to $\mathcal{S}_{t-1} \cap \{\pi_\theta(\cdot) \geq \rho\}$,

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;\approx\; -\mathbb{E}_{x\sim\pi_\theta \,|\, \mathcal{S}_{t-1}}[\log \pi_\theta(x)].$$

This is the negative log-likelihood of π_θ under itself, the defining form of a self-distillation step. The restriction matters. Without it, samples from π_θ itself would give a zero expected gradient. Sampling at temperature 0.7 with top-p 0.95 and filtering by the verifier both cut off low-probability outputs. The update therefore moves probability mass toward the high-probability region.

Conclusion. Iterating across rounds $t \geq T$ corresponds to repeated minimisation of this self-referential loss. By the analysis of Mobahi et al. (2020), each such step shrinks the effective hypothesis space of π_θ (in the Hilbert-space regularisation sense), progressively suppressing low-probability branches. Phase III's contraction is the repeated demotion of correct but low-probability solution paths. This is the corresponding mechanism in our setting. □

The main experiments train with GRPO rather than SFT. Samples with positive advantage enter the GRPO gradient as weighted log-likelihood terms, so the argument above applies to them with sample weights.

J DIRECTIONAL INFORMATION GAIN AND EXTERNAL-PHASE TRAINING

This appendix describes an optional extension of the closed-loop framework of Section 3. It continues training on samples drawn from outside the loop after the closed-loop C_t has saturated. The extension introduces a directional counterpart to C_t and uses it with c_t to select external samples in a second training phase.

Why a directional gain is needed. Section 3.3 shows that continuing closed-loop training while $C_t \to 0$ drives the main model into the contraction of Phase III. To keep improving, the model must take in samples from outside the closed loop. However, C_t alone is insufficient for selecting such samples. A positive $c_t(e)$ only indicates that e lies outside $\mathcal{D}_{t-1}$, not that it addresses the model's current weakness. A sample outside $\mathcal{D}_{t-1}$ that the model already handles correctly merely reinforces existing capability when added to training. We therefore construct a complementary criterion that identifies whether a sample targets that weakness.

Definition. Partition $\mathcal{D}_t$ by the verifier into a correct part $\mathcal{D}_t^+$ and an incorrect part $\mathcal{D}_t^-$, and fit reference distributions P_t^+ on $\mathcal{D}_t^+$ and P_t^- on $\mathcal{D}_t^-$. The two thus capture the current correct and incorrect behavior of the model. For an external sample e, the **directional learnable information gain** is

$$C_t^d(e) = \ell(e; P_t^+) - \ell(e; P_t^-). \tag{28}$$

$C_t^d(e) > 0$ indicates that the encoding cost of e under P_t^- is lower than under P_t^+, that is, e is more similar to the samples the model gets wrong. The model performs poorly on these, so adding e to training targets its weakness. $C_t^d(e) < 0$ indicates the opposite. Then e is more similar to the samples the model already handles, and adding it merely reinforces existing capability. The two scores therefore serve as independent criteria. The first controls whether e is novel relative to $\mathcal{D}_{t-1}$. The second controls whether e targets the model's weakness. Samples positive on both pass the dual gate of the external phase.

Estimation and selection. We implement P_t^+ and P_t^- as proxies M_t^+ and M_t^-, fine-tuned on $\mathcal{D}_t^+$ and $\mathcal{D}_t^-$ at the start of the external phase under the protocol of Section 3.1.2. $C_t^d(e)$ is then the encoding-cost gap between the two proxies on e,

$$C_t^d(e) = \ell_{M_t^+}(e) - \ell_{M_t^-}(e). \tag{29}$$

Like $c_t(e)$, $C_t^d(e)$ is computed once per candidate e in the external pool E before training. A candidate enters training only when both $c_t(e)$ and $C_t^d(e)$ are positive.

External-phase procedure. The closed-loop phase of Section 3.3 runs until C_t falls below the threshold τ for two consecutive rounds. At the stopping round, $\mathcal{D}_t$ is partitioned by the verifier into $\mathcal{D}_t^+$ and $\mathcal{D}_t^-$, and the proxies M_t^+ and M_t^- are fine-tuned following the protocol of Section 3.1.2. From this round onward, training samples are drawn from the external pool E rather than from the closed-loop $\mathcal{D}_t$. For each candidate $e \in E$, both $c_t(e)$ and $C_t^d(e)$ are computed, and only candidates that pass the dual gate enter training. The external phase terminates when C_t and C_t^d

remain below threshold simultaneously over an extended period, indicating that the external pool no longer contributes training signal under either criterion.

Other external sources. The external pool is not restricted to static corpora. Samples built from tool feedback or human feedback enter it as ordinary candidates. Both $c_t(e)$ and $C_t^d(e)$ depend only on the sample text, so the same dual gate applies to any source.

K Limitations and Broader Impact

Limitations. ATRI relies on three assumptions whose violation is worth flagging. (i) *Closed-loop structure.* The diagnostic $C_t = f(\mathcal{D}_t, \mathcal{D}_{t-1})$ assumes the existence of a well-defined per-round training set. Appendix H replaces rounds by fixed token windows and finds that the decision moves by at most one checkpoint. Settings without round boundaries therefore still require a window size to be chosen. (ii) *Proxy approximation.* C_t is computed through a finite-capacity proxy language model rather than the true previous-round distribution p_{t-1}. The sensitivity sweep in Table 3 shows this is robust within the standard range, but pathological dataset distributions (e.g. extreme length skew) could in principle break the proxy's calibration. We have not encountered this in practice. (iii) *External pool quality.* The external phase assumes access to an external candidate pool E that contains samples beyond $\mathcal{D}_t$ in some directions. When such samples are unavailable (e.g. in fully novel domains where no external data exists), the external phase has nothing to add. Training then ends at the stopping round, and ATRI's gain comes from the closed-loop control alone.

A further limitation concerns *verifier reliability*. The ATRI loop uses majority-vote pseudo-labels, and the Vanilla analysis of Section 2 uses ground-truth answer matching. Neither involves a learned reward model. In deployments where the verifier is itself a learned model, classical reward-hacking failure modes can resurface. The alarm of C_t may be less affected, since C_t depends on the dataset-level distribution of $\mathcal{D}_t$ rather than on per-sample verifier labels. This setting is not tested here. The absolute accuracy ceiling is still verifier-bounded.

Finally, the threshold $\tau = 0.1\, C_1$ is an empirical choice whose insensitivity is established only over the range $\alpha \in [0.05, 0.30]$ (Table 3). The constant 0.1 reflects a trade-off rather than a derived optimum.

The theoretical statements hold under stated assumptions. Corollary 1 requires the proxy to fit the previous round exactly. The monotone decline of Proposition 3 requires the contraction condition. External data can break this condition. The lifecycle itself is an empirical regularity rather than a theorem.

Our evidence for the lifecycle covers Qwen and Llama base models, MATH and MBPP tasks, Pythia and OPT proxies, and six training algorithms. Claims beyond these settings remain empirical extrapolation.

Compute resources. All experiments are run on internal GPU clusters of NVIDIA H800 80GB and A100 80GB nodes. Each main-comparison run uses a single 8-GPU node. The Pythia-160M proxy fitting and per-sample C_t computation at each round are negligible relative to the main fine-tuning cost. Per-round PFLOPs for the C_t control layer are reported in Table 5, and the proxy-capacity rationale is detailed in Appendix D. Table 11 adds measured per-round GPU-hours at three base-model scales. The main loop grows with the base model. The C_t control layer stays near 0.35 GPU-hours because the proxy size is fixed. The relative overhead therefore falls from 4.1% at 4B to 1.3% at 14B.

Table 11: Measured per-round cost in GPU-hours at three base-model scales.

Base model	Main loop	Proxy + C_t	Relative overhead
Qwen3-4B-Base	8.2	0.35	4.1%
Qwen3-8B-Base	15.5	0.35	2.3%
Qwen3-14B-Base	26.8	0.35	1.3%

Broader impact. The most direct consequence of C_t-based control is a reduction in wasted compute. Terminating self-evolution at the diagnosed saturation point avoids post-peak training. In the 15-round runs of Figure 5, the six baselines train for 6 to 11 rounds after their accuracy peak. The saved compute can be left unspent or moved to the external phase.

A second consequence is methodological. C_t provides a unified diagnostic that can be adopted on top of any existing self-evolution algorithm without re-engineering. It lowers the cost of comparing methods and encourages more honest reporting of long-horizon trajectories rather than peak-accuracy snapshots.

Like any technique that improves the training efficiency of LLMs, ATRI accelerates the underlying capability trajectory of self-evolving systems. The accompanying considerations are familiar from the broader self-improvement literature. More efficient autonomous training raises the importance of robust verifiers, of evaluation suites that probe generation diversity rather than only single-sample accuracy, and of alignment work that scales with capability. Our work contributes a diagnostic tool rather than a new capability vector. Practitioners adopting it should also report dimensions beyond accuracy, such as pass@k, behavioral diversity, and out-of-distribution behavior.