# SOCRATIC-ZERO: BOOTSTRAPPING REASONING VIA DATA-FREE AGENT CO-EVOLUTION

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

## ABSTRACT

Recent breakthroughs in large language models (LLMs) on reasoning tasks rely heavily on massive, high-quality datasets–typically human-annotated and thus difficult to scale. While data synthesis or distillation offers a promising alternative, existing methods struggle with inconsistent data quality and an inability to dynamically adapt to the evolving capabilities of the model, leading to suboptimal training signals. To address these limitations, we introduce *Socratic-Zero*, a fully autonomous framework that generates high-quality training data from minimal seed examples through the co-evolution of three agents: the *Teacher*, the *Solver*, and the *Generator*. The *Solver* continuously refines its reasoning by learning from preference feedback on both successful and failed trajectories; the *Teacher* adaptively crafts increasingly challenging questions based on the Solver's weaknesses; and the *Generator* distills the Teacher's question-design strategy to enable scalable, high-fidelity curriculum generation. This closed-loop system produces a self-improving curriculum without requiring large pre-existing task collections or human annotations. After a one-time cold-start SFT on 1,500 original MATH Level-5 problem-solution pairs, co-evolution starts from 100 curated seed questions and yields an average gain of +20.2 percentage points on seven mathematical reasoning benchmarks (AMC23, AIME24-25, Olympiad, MATH-500, Minerva, and GSM8K). We also evaluate selected non-mathematical domains, including science reasoning, deep research, and software engineering. In the corrected generator comparison, Socratic-Generator-32B reaches 36.04% average downstream utility; a separate 10K-problem, five-run evaluation reaches $39.61\% \pm 0.17$, compared with $38.79\% \pm 0.23$ for the strongest commercial generator.

## 1 INTRODUCTION

The pursuit of advanced mathematical reasoning in large language models has reached a critical juncture. While recent breakthroughs have demonstrated remarkable capabilities on complex mathematical problems (Hendrycks et al., 2021; Cobbe et al., 2021), these advances rely on massive datasets of meticulously curated reasoning trajectories – a requirement that is both costly and fundamentally unscalable. Current state-of-the-art models depend on millions of human-annotated problem-solution pairs and hand-designed curricula (Yu et al., 2024), creating a fundamental bottleneck that limits both accessibility and the potential for models to evolve beyond human-curated knowledge boundaries.

Current methodologies remain entrenched in a static paradigm: datasets are frozen upon collection, curricula are handcrafted in advance, and models are trained on fixed problem distributions. This approach suffers from critical weaknesses: it cannot adapt to evolving model capabilities during training, fails to exploit rich feedback signals for targeting specific weaknesses, and requires extensive human expertise for curriculum design. Recent efforts through synthetic data generation (Lee et al., 2024; Chen et al., 2025b) and iterative training (Zhao et al., 2025; Huang et al., 2025b) have shown promise but remain constrained by their reliance on external supervision and lack of effective quality control mechanisms for synthesized content.

To overcome these limitations, we introduce **Socratic-Zero**, a paradigm-shifting framework that eliminates dependency on large-scale external datasets while enabling truly autonomous reasoning improvement. Inspired by the Socratic method of learning through questioning (Figure 1(a)), our



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

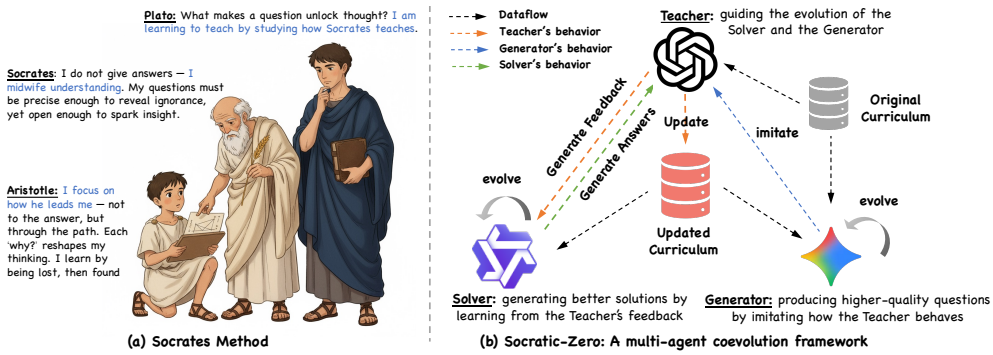


Figure 1: (a) The Socratic method of teaching through targeted questioning. (b) Our Socratic-Zero framework operationalizes this idea as a loop among a Solver, a Teacher, and a Generator.

approach implements co-evolution between three agents: a *Solver* that attempts to solve mathematical questions, a *Teacher* that strategically generates challenging problems to expose the Solver's weaknesses, and a *Generator* that learns to distill and scale the Teacher's problem generation strategy. This architecture (Figure 1(b)) translates the philosophical dialogue of the Socratic method into a concrete, co-evolutionary computational framework. Unlike conventional pipelines that decouple data generation from model training, Socratic-Zero unifies them within a continuous co-evolutionary loop formalized as a optimization problem. Our contributions are threefold:

- **Multi-Agent Co-Evolutionary Framework:** We establish a theoretical foundation for co-evolutionary learning where the Solver, Teacher, and Generator agents interact dynamically, formalizing reasoning improvement as an adaptive curriculum learning problem (Figure 1(b)).
- **Socratic-Zero System:** We implement a concrete framework where the Solver improves via preference learning, the Teacher evaluates correctness and generates adaptive curriculum, and the Generator learns strategic distillation through value-weighted supervised fine-tuning (WSFT), enabling autonomous reasoning advancement from minimal seed data.
- **Superior Empirical Performance:** Our Socratic-Solver-8B achieves +20.2 points average improvement across seven mathematical reasoning benchmarks. In the corrected seven-task generator comparison, Socratic-Generator-32B reaches 36.04% average downstream utility; a separate 10K-problem, five-run evaluation reaches 39.61% $\pm$ 0.17, above the strongest commercial generator at 38.79% $\pm$ 0.23.

## 2 Related Work

**Data Synthesis.** To alleviate data scarcity, researchers have leveraged LLMs' generative capabilities to synthesize training samples. Early approaches used prompt engineering to guide question-answer generation (Yu et al., 2024; Zhan et al., 2025). Subsequently, LLM2LLM (Lee et al., 2024) and WarriorMath (Chen et al., 2025b) introduced deficiency-aware mechanisms, where teacher models identify knowledge gaps and generate targeted data. More recently, Absolute Zero (Zhao et al., 2025) and R-Zero (Huang et al., 2025b) explored fully autonomous self-play paradigms for continuous task generation and learning. While these advances achieve data autonomy, they lack effective quality control mechanisms, resulting in repeated use of low-value samples that severely impact effectiveness.

**Data Distillation.** Knowledge distillation transfers capabilities from powerful teacher models to lighter student models. Early work like Orca (Mukherjee et al., 2023) used imitation learning to replicate teacher reasoning chains. Policy distillation (Xu et al., 2025) extends this by transferring dynamic decision-making strategies. GKD (Agarwal et al., 2024) enables students to learn from their own sequences using teacher feedback for policy correction. However, students passively accept teacher feedback without evaluating reliability, degrading learning quality when guidance is suboptimal. These methods also rely on static datasets, unable to dynamically adjust content based on students' evolving capabilities. While recent advances (Wang et al., 2025; Sachdeva & McAuley,

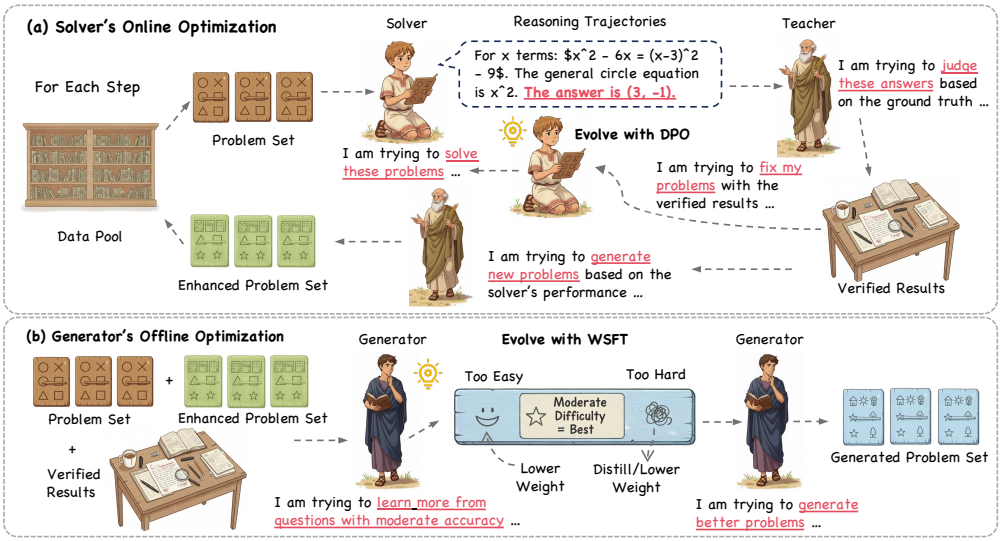


Figure 2: Overview of the Socratic-Zero Framework. (a) Solver Evolving: The Solver attempts to solve problems and learns from preference pairs of correct and incorrect solutions via DPO, while the frozen Teacher strategically generates challenging problems based on Solver failures using fixed generation and evaluation functions. (b) Generator Evolving: The Generator distills the Teacher's problem generation strategy using value-weighted supervised learning. Together, these create a self-improving loop where the curriculum dynamically evolves to maintain optimal challenge levels for the Solver's current capabilities.

2023; Maekawa et al., 2024; Ko et al., 2024; Liu et al., 2025) promote data-centric optimization, they lack effective quality control and adaptive curriculum generation.

**Preference Learning.** Translating feedback signals into model optimization is central to self-evolution systems. Early approaches like RLHF (Stiennon et al., 2020) trained reward models on human preferences then fine-tuned policies, but this process is complex and unstable. Recent methods like DPO (Rafailov et al., 2023) and RWSFT (Mukherjee et al., 2025) directly optimize preferences, improving efficiency and stability. Combined with self-correction mechanisms like Self-Refine (Madaan et al., 2023), models possess preliminary closed-loop capabilities. Further advances including Self-Evolved Reward Learning (Huang et al., 2025a), Self-Play Fine-Tuning (Chen et al., 2024), and Self-Play Critic (Chen et al., 2025a) explore autonomous feedback strategies. However, these methods lack unified, co-evolving frameworks for feedback generation and validation.

# 3 Methodology

## 3.1 The Socratic-Zero Framework

We introduce **Socratic-Zero**, a fully autonomous, co-evolutionary framework designed to bootstrap mathematical reasoning from a minimal set of seed problems, entirely without relying on external human-annotated data. As illustrated in Figure 2, the system operates as a self-improving loop among three agents: a **Solver** that learns to reason, a fixed **Teacher** that acts as an oracle for evaluation and problem refinement, and a **Generator** that learns to synthesize a curriculum.

At each iteration $t$, the framework operates on a curriculum of problems and their reference solutions, denoted as $\mathcal{D}_t = \{(q, y_{\text{ref}})\}$. The core of Socratic-Zero is the co-evolution of the Solver and Generator under the guidance of the Teacher. The Solver is trained to solve problems from $\mathcal{D}_t$. Its failures are then used by the Teacher to create new, targeted problems. The Generator, in turn, distills the Teacher's strategy to produce a scalable curriculum. This process creates a dynamic curriculum that continuously adapts to the Solver's evolving capabilities, ensuring the training signals remain maximally informative. The agents are formally defined as follows:

**Agents in the Socratic-Zero Framework**

1. **Solver ($\mathcal{S}$):** An agent with a policy $\pi_{\theta_\mathcal{S}}$, parameterized by $\theta_\mathcal{S}$, which maps a problem $q$ to a solution trajectory $y$. Its objective is to generate correct reasoning paths. At each iteration $t$, it improves by learning from preference feedback on its own attempts.
2. **Teacher ($\mathcal{T}$):** A fixed, high-capacity LLM that provides two deterministic oracle functions: (i) a verification function $V(q, y) \to \{0, 1\}$, which judges the correctness of a solution $y$ for a problem $q$; and (ii) a problem refinement function $G(q, y_{\text{fail}}) \to (q', y'_{\text{ref}})$, which creates a new problem-solution pair by revising an original problem $q$ based on a failed solution $y_{\text{fail}}$.
3. **Generator ($\mathcal{G}$):** An agent with a policy $\pi_{\theta_\mathcal{G}}$, parameterized by $\theta_\mathcal{G}$, that learns to mimic the Teacher's refinement strategy. It maps a problem and a failed solution $(q, y_{\text{fail}})$ to a new problem $q'$. It evolves to generate problems that are optimally challenging for the current Solver.

The curriculum expands based on the Solver's mistakes. At iteration $t$, the set of Solver failures on the current curriculum $\mathcal{D}_t$ is collected:

$$\mathcal{F}_t = \big\{(q, y_\mathcal{S}) \,|(q, y_{\text{ref}}) \in \mathcal{D}_t,\ y_\mathcal{S} \sim \pi_{\theta_\mathcal{S}^{(t)}}(\cdot \mid q), \\ V(q, y_\mathcal{S}) = 0\big\}. \tag{1}$$

The Teacher refines each failure into a new, instructive problem-solution pair. The set of these new pairs, $\mathcal{D}_{\text{new}}$, is used to augment the curriculum for the next iteration:

$$\mathcal{D}_{t+1} = \mathcal{D}_t \cup \underbrace{\{G(q, y_\mathcal{S}) \mid (q, y_\mathcal{S}) \in \mathcal{F}_t\}}_{\mathcal{D}_{\text{new}}}. \tag{2}$$

The Solver and the Generator co-evolve under the Teacher's guidance. On one hand, the Solver minimizes its expected solution loss:

$$\mathcal{L}_\mathcal{S}(\mathcal{D}_t, \pi^{(t)}_{\theta_\mathcal{S}}) = \mathbb{E}_{q\sim\mathcal{D}_t}\big[\ell_\mathcal{S}(\pi^{(t)}_{\theta_\mathcal{S}}(q), q)\big]. \tag{3}$$

On the other hand, the Generator minimizes its expected generation loss:

$$\mathcal{L}_\mathcal{G}(\mathcal{D}_t, \pi^{(t)}_{\theta_\mathcal{G}}, U) = \mathbb{E}_{q\sim\mathcal{D}_t}\big[\ell_\mathcal{G}(\pi^{(t)}_{\theta_\mathcal{G}}(q), q, q_\mathcal{T}, U)\big]. \tag{4}$$

The full co-evolutionary training procedure is detailed in Algorithm 1.

### 3.2 Solver Training via Online Preference Optimization

The Solver's policy $\pi_{\theta_\mathcal{S}}$ is improved through online preference learning, leveraging the Teacher's verification function $V$ to create a feedback loop. For each problem $q \in \mathcal{D}_t$, the Solver generates $k$ solution attempts, $\{y^{(i)}_\mathcal{S}\}_{i=1}^k$. These attempts are partitioned into a set of "winning" (correct) solutions $\mathcal{Y}_w(q)$ and "losing" (incorrect) solutions $\mathcal{Y}_l(q)$:

$$\begin{cases} \mathcal{Y}_l(q) = \{y^{(i)}_\mathcal{S} \mid V(q, y^{(i)}_\mathcal{S}) = 0\}, \\ \mathcal{Y}_w(q) = \{y^{(i)}_\mathcal{S} \mid V(q, y^{(i)}_\mathcal{S}) = 1\} \\ \qquad\qquad \cup \left\{y_{\text{ref}} \,\middle|\, \forall i,\ V(q, y^{(i)}_\mathcal{S}) = 0\right\}. \end{cases} \tag{5}$$

where $y_{\text{ref}}$ is the reference solution from the curriculum. If the Solver fails to generate any correct solution, the ground-truth solution serves as the sole winning example. This ensures a valid preference pair $(y_w, y_l)$, where $y_w \in \mathcal{Y}_w(q)$ and $y_l \in \mathcal{Y}_l(q)$, can always be constructed.

The Solver's parameters $\theta_\mathcal{S}$ are then updated using the Direct Preference Optimization (DPO) loss (Rafailov et al., 2023). This objective maximizes the likelihood of preferred solutions over rejected ones:

$$\mathcal{L}_{\text{DPO}}(\theta_\mathcal{S}; \theta_{\text{ref}}) = -\mathbb{E}_{\substack{q\sim\mathcal{D}_t, \\ y_w\sim\mathcal{Y}_w(q), \\ y_l\sim\mathcal{Y}_l(q)}}\left[\log\sigma\left(\beta\log\frac{\pi_{\theta_\mathcal{S}}(y_w \mid q)}{\pi_{\theta_{\text{ref}}}(y_w \mid q)} - \beta\log\frac{\pi_{\theta_\mathcal{S}}(y_l \mid q)}{\pi_{\theta_{\text{ref}}}(y_l \mid q)}\right)\right], \tag{6}$$

where $\pi_{\theta_{\text{ref}}}$ is a frozen reference policy (e.g., $\pi_{\theta_\mathcal{S}}$ from the start of the iteration), $\beta$ is a temperature hyperparameter, and $\sigma$ is the sigmoid function.

### 3.3 Generator Training via Offline Value-Weighted Distillation

To ensure scalable curriculum generation without perpetual reliance on the expensive Teacher, the Generator $\pi_{\theta_\mathcal{G}}$ is trained to distill the Teacher's problem refinement strategy. An effective curriculum should feature problems of *desirable difficulty*–challenging enough to be informative but not so difficult as to be unsolvable.

We formalize this concept with a utility function $U(q'|\pi_{\theta_\mathcal{S}})$ that scores a new problem $q'$ based on the current Solver's performance. Let $s_q = \frac{1}{k}\sum_{i=1}^{k} V(q, y_\mathcal{S}^{(i)})$ be the success rate of the Solver $\pi_{\theta_\mathcal{S}}$ over $k$ attempts. The utility is defined by an unnormalized Gaussian centered at a target success rate $\mu$:

$$U(q'|\pi_{\theta_\mathcal{S}}) = \exp\left(-\frac{(s_{q'}-\mu)^2}{2\sigma^2}\right). \tag{7}$$

For consistency with our value-based formulation, we also denote this utility as $v(q') \equiv U(q'|\pi_{\theta_\mathcal{S}})$:

$$v(q') = \exp\left(-\frac{(s_{q'}-\mu)^2}{2\sigma^2}\right). \tag{8}$$

We set $\mu = 0.5$ to incentivize problems at the frontier of the Solver's capabilities, with $\sigma$ controlling the tolerance for deviation.

The Generator is trained via weighted supervised fine-tuning (WSFT) to mimic the Teacher's generation of high-utility problems. The training data $\mathcal{D}_\mathcal{G}$ is constructed from the set of Solver failures $\mathcal{F}_t$ and the corresponding Teacher-refined problems from $\mathcal{D}_{\text{new}}$:

$$\mathcal{D}_\mathcal{G} = \left\{(q, y_{\text{fail}}, q') \mid (q, y_{\text{fail}}) \in \mathcal{F}_t,\ (q', y'_{\text{ref}}) = G(q, y_{\text{fail}})\right\}. \tag{9}$$

The Generator's objective is to maximize the utility-weighted log-likelihood of producing the Teacher's refined problems:

$$\mathcal{L}_{\text{WSFT}}(\theta_\mathcal{G}) = -\mathbb{E}_{(q,y_{\text{fail}},q')\sim\mathcal{D}_\mathcal{G}}\left[U(q' \mid \pi_{\theta_\mathcal{S}}) \cdot \log \pi_{\theta_\mathcal{G}}(q' \mid q, y_{\text{fail}})\right]. \tag{10}$$

This objective steers the Generator towards producing problems that are optimally challenging for the current Solver, effectively internalizing the Teacher's expert curriculum design principles.

## 4 Experiments

### 4.1 Experimental Setup

**Experimental Setup.** We employed Qwen3-235B-A22B as the Teacher, Qwen3-32B as the Generator, and conducted Solver experiments on Qwen3-8B-base. The base Solver first receives one-time SFT on 1,500 original MATH Level-5 problem-solution pairs. Co-evolution then starts from a separate set of 100 curated MATH seed problems and produces approximately 7.9K training examples for the main Stage-3 result. We evaluated on seven mathematical reasoning benchmarks (AMC, Minerva, MATH-500, GSM8K, Olympiad-Bench, AIME-2024, AIME-2025) and three general reasoning benchmarks (BBEH, MMLU-Pro, SuperGPQA). Complete experimental details are provided in the appendix.

**Baseline Comparison.** Table 1 shows Socratic-Zero achieves 56.1% average accuracy. The corrected averages for Absolute Zero, R-Zero, and SPICE are 52.6%, 54.7%, and 53.8%, respectively; Socratic-Zero remains highest while using approximately 7.9K co-evolution examples.

**Generalization Beyond Mathematical Reasoning.** We further examine whether Socratic-Zero generalizes beyond mathematical reasoning. Table 2 shows that our framework consistently improves performance across diverse domains, including science reasoning, deep research, software engineering, multimodal chart reasoning, and multimodal coding. Notably, the improvements are observed across both text-only and vision-language models, indicating that the benefit of Socratic-Zero is not restricted to a specific task type or model family.

This broad transfer suggests that the capabilities learned through Socratic-Zero are not narrowly specialized to mathematical benchmarks, but instead reflect more general reasoning and problem-solving skills. The results support our central claim that co-evolutionary data generation can serve as a scalable mechanism for improving model reasoning in domains where high-quality supervision is scarce or expensive to obtain.

Table 1: Solver evaluation results with different training methods. Results are reported on seven benchmarks (AMC, Minerva, MATH-500, GSM8K, Olympiad, AIME-25, AIME-24) and their average. Data Scale denotes the amount of external or generated training data used by each method. Arrow values represent absolute point changes relative to Initial SFT, where ↑ indicates improvement and ↓ indicates decline. For methods whose per-benchmark breakdown is not explicitly reported, we provide approximate estimates for completeness.

| Training Method | Data Scale | Benchmark Datasets | | | | | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| | | AMC | Minerva | MATH-500 | GSM8K | Olympiad | AIME-25 | AIME-24 | |
| *Qwen3-8B-base* | | | | | | | | | |
| Base Model | – | 32.5 | 31.3 | 48.8 | 63.4 | 24.1 | 4.2 | 5.1 | 29.9 |
| Initial SFT | – | 39.1 | 37.8 | 56.9 | 68.2 | 31.7 | 8.1 | 9.3 | 35.9 |
| Static Augmentation | ∼32K | 45.8 ↑6.7 | 41.9 ↑4.1 | 62.7 ↑5.8 | 74.6 ↑6.4 | 35.9 ↑4.2 | 11.4 ↑3.3 | 12.3 ↑3.0 | 40.7 ↑4.8 |
| Distillation (memath_math) | ∼75K | 51.2 ↑12.1 | 44.1 ↑6.3 | 70.3 ↑13.4 | 78.5 ↑10.3 | 38.4 ↑6.7 | 12.2 ↑4.1 | 14.9 ↑5.6 | 44.3 ↑8.4 |
| DPO (Polaris) | ∼53K | 54.7 ↑15.6 | 47.2 ↑9.4 | 71.8 ↑14.9 | 83.5 ↑15.3 | 44.6 ↑12.9 | 17.8 ↑9.7 | 19.8 ↑10.5 | 48.4 ↑12.5 |
| DPO (MATH-full) | ∼7.5K | 48.9 ↑9.8 | 43.6 ↑5.8 | 65.3 ↑8.4 | 77.8 ↑9.6 | 38.7 ↑7.0 | 13.1 ↑5.0 | 16.0 ↑6.7 | 43.1 ↑7.2 |
| LLM2LLM | ∼16K | 44.9 ↑5.8 | 42.1 ↑4.3 | 66.8 ↑9.9 | 79.4 ↑11.2 | 34.6 ↑2.9 | 7.9 ↓0.2 | 10.4 ↑1.1 | 40.9 ↑5.0 |
| Absolute Zero | ∼40K | 62.5 ↑23.4 | 52.9 ↑15.1 | 76.6 ↑19.7 | 92.0 ↑23.8 | 47.8 ↑16.1 | 18.2 ↑10.1 | 18.4 ↑9.1 | 52.6 ↑16.7 |
| SPIRAL | ∼51K | 57.5 ↑18.4 | 42.4 ↑4.6 | 76.4 ↑19.5 | 91.0 ↑22.8 | 38.4 ↑6.7 | 10.0 ↑1.9 | 13.3 ↑4.0 | 47.0 ↑11.1 |
| R-Zero | ∼24K | 61.7 ↑22.6 | 60.7 ↑22.9 | 82.0 ↑25.1 | 94.1 ↑25.9 | 48.9 ↑17.2 | 19.2 ↑11.1 | 16.4 ↑7.1 | 54.7 ↑18.8 |
| SPICE | ∼82K | 60.8 ↑21.7 | 50.6 ↑12.8 | 80.1 ↑23.2 | 93.2 ↑25.0 | 50.4 ↑18.7 | 21.7 ↑13.6 | 20.1 ↑10.8 | 53.8 ↑17.9 |
| **Socratic-Zero (Ours)** | ∼7.9K | **63.7** ↑24.6 | **52.4** ↑14.6 | **81.2** ↑24.3 | **87.3** ↑19.1 | **55.1** ↑23.4 | **24.6** ↑16.5 | **28.4** ↑19.1 | **56.1** ↑20.2 |

Table 2: Cross-domain and cross-modality generalization of Socratic-Zero. We evaluate our framework beyond mathematical reasoning on science reasoning, deep research, software engineering, multimodal chart understanding, and multimodal coding.

| Domain | Model | Benchmark | Base | Ours |
|---|---|---|---|---|
| *Science Reasoning* | Qwen3-8B-Base | MMLU-Pro | 50.0 | **65.8** ↑15.8 |
| | | GPQA | 41.2 | **56.6** ↑15.4 |
| | | SuperGPQA | 24.7 | **39.8** ↑15.1 |
| *Deep Research* | Qwen2.5-7B-Instruct | NQ | 13.0 | **49.0** ↑36.0 |
| | | TriviaQA | 41.0 | **71.0** ↑30.0 |
| | | PopQA | 14.0 | **50.0** ↑36.0 |
| | | HotpotQA | 18.0 | **46.0** ↑28.0 |
| | | 2WikiMQA | 25.0 | **65.0** ↑40.0 |
| | | MuSiQue | 3.0 | **15.0** ↑12.0 |
| | | Bamboogle | 12.0 | **42.0** ↑30.0 |
| *Software Engineering* | Qwen3-4B-Instruct-2507 | SWE-bench Verified | 21.6 | **30.0** ↑8.4 |
| *Multimodal Chart Reasoning* | Qwen2.5-VL-7B-Instruct | ChartQA | 87.3 | **91.2** ↑3.9 |
| | | CharXiv | 66.6 | **74.2** ↑7.6 |
| | | ChartQAPro | 41.3 | **46.3** ↑5.0 |
| | | ChartMiniC | 40.2 | **45.3** ↑5.1 |
| *Multimodal Coding* | Qwen2.5-VL-7B-Instruct | Design2Code | 29.1 | **34.3** ↑5.2 |
| | | UIFlow2Code | 75.9 | **81.5** ↑5.6 |

### 4.2 Generator Results

We assessed both the intrinsic quality of generated problems and their downstream training effectiveness. The Generator's ability to produce high-quality, appropriately challenging problems is crucial for the success of our co-evolutionary framework.

#### 4.2.1 Problem Quality Assessment

We first evaluate the validity of generated problems by having human annotators assess whether each generated problem is mathematically well-formed, has a clear question, and admits a definite answer. As shown in Table 3, our specialized Socratic-Generator-32B achieves a remarkable 95.6% validity rate, rivaling significantly larger models. This high validity rate is particularly impressive given that our Generator is over 20× smaller than some of the commercial models it matches or exceeds.

Table 3: Generator validity rates across different models. Validity rate is measured as the percentage of generated problems successfully solved by Qwen3-235B.

| GENERATOR MODEL | VALIDITY RATE (%) |
|---|---|
| QWEN3-32B | 89.1 |
| QWEN3-235B-A22B | 95.1 ↑6.0 |
| GEMINI-2.5-PRO | 94.2 ↑5.1 |
| GPT5-GLOBAL | 95.8 ↑6.7 |
| DEEPSEEK-V3.1-671B | 96.5 ↑7.4 |
| GROK4 | 95.7 ↑6.7 |
| CLAUDE-4.1-OPUS | 96.9 ↑7.8 |
| **SOCRATIC-GENERATOR-32B** | **95.6** ↑6.5 |

Table 4: Downstream training effectiveness of data generated by different generator models. Results are reported on seven benchmarks (AIME-24, AIME-25, AMC-23, GSM8K, MATH-500, Minerva, Olympiad) and their average. Values with arrows represent absolute point changes relative to baseline.

| | BENCHMARK DATASETS | | | | | | | AVG. |
|---|---|---|---|---|---|---|---|---|
| | AIME-24 | AIME-25 | AMC-23 | GSM8K | MATH-500 | MINERVA | OLYMPIAD | |
| *DeepSeek-R1-Distill-Llama-8B* | | | | | | | | |
| + ZERO-SHOT | 5.8 | 8.3 | 42.5 | 72.2 | 52.4 | 15.3 | 23.0 | 32.75 |
| *Open-Sourced Generators* | | | | | | | | |
| QWEN3-32B | 9.2 | 10.0 | 44.4 | 75.7 | 55.7 | 15.1 | 24.5 | 33.51 ↑0.76 |
| QWEN3-235B-A22B-INSTRUCT-2507 | 12.5 ↑3.3 | 12.5 ↑2.5 | 47.5 ↑3.1 | 76.1 ↑0.4 | 57.8 ↑2.1 | 16.4 ↑1.3 | 23.6 ↓0.9 | 35.20 ↑2.45 |
| DEEPSEEK-V3.1-671B | 12.5 ↑3.3 | 11.7 ↑1.7 | 46.2 ↑1.8 | 76.4 ↑0.7 | 56.4 ↑0.7 | 16.5 ↑1.4 | 23.9 ↓0.6 | 34.80 ↑2.05 |
| *Advanced Commercial Generators* | | | | | | | | |
| GEMINI-2.5-PRO-06-17 | 10.0 ↑0.8 | 15.0 ↑5.0 | 46.9 ↑2.5 | 78.1 ↑2.4 | 57.2 ↑1.5 | 16.0 ↑0.9 | 25.4 ↑0.9 | 35.51 ↑2.76 |
| GPT5-0807-GLOBAL | 12.5 ↑3.3 | 13.3 ↑3.3 | 45.0 ↑0.6 | 76.8 ↑1.1 | 56.6 ↑0.9 | 15.5 ↑0.4 | 25.9 ↑1.4 | 35.09 ↑2.34 |
| GROK-4 | 11.7 ↑2.5 | 12.5 ↑2.5 | 45.8 ↑1.4 | 76.3 ↑0.6 | 56.9 ↑1.2 | 15.9 ↑0.8 | 24.9 ↑0.4 | 34.86 ↑2.11 |
| CLAUDE-4.1-OPUS | 13.3 ↑4.1 | 13.8 ↑3.8 | 46.5 ↑2.1 | 77.3 ↑1.6 | 57.5 ↑1.8 | 16.7 ↑1.6 | 24.3 ↓0.2 | 35.63 ↑2.88 |
| **SOCRATIC-GENERATOR-32B** | **12.5** ↑3.3 | **13.3** ↑3.3 | **48.1** ↑3.7 | **77.6** ↑1.9 | **57.8** ↑2.1 | **18.4** ↑3.3 | **24.6** ↑0.1 | **36.04** ↑3.29 |

The strong performance of our Generator demonstrates that the value-weighted distillation approach effectively captures the Teacher's problem generation strategy. By focusing on problems with optimal difficulty (success rate near 0.5), the Generator learns to produce problems that are both valid and appropriately challenging.

#### 4.2.2 DOWNSTREAM EFFECTIVENESS

While validity is important, the ultimate test of a Generator is whether its synthetic data improves downstream model performance. Table 4 reports the corrected seven-task utility; Socratic-Generator-32B reaches 36.04%. In a separate 10K-problem evaluation with five independent student runs, it reaches 39.61% $\pm$ 0.17 versus 38.79% $\pm$ 0.23 for the strongest commercial generator.

#### 4.2.3 GENERATOR-AS-TEACHER

A central goal of Socratic-Zero is to remove long-term dependence on an expensive external Teacher by distilling its curriculum-generation ability into a smaller Generator. To evaluate whether the Generator can take over this role, we start from a converged Solver checkpoint with 56.1% average accuracy at $\sim$8K cumulative training data, and continue training under two curriculum sources: the original Teacher (Qwen3-235B-A22B-Instruct-2507) and our Socratic-Generator-32B.

Table 5 shows that Generator-guided training reaches **60.5%** average accuracy at $\sim$16K cumulative data, versus **60.3%** for Teacher guidance, with gains on Minerva (+0.7pp) and AIME-24 (+0.7pp). At $\sim$28K and $\sim$47K, the Generator reaches **64.1%** and **67.5%**, compared with **62.3%** and **62.8%** under Teacher guidance. The Generator therefore provides a practical curriculum source for sustained autonomous co-evolution (Appendix N).

Table 5: Comparison between Teacher-guided and Generator-guided continued Solver training under increasing curriculum data scales. Both settings start from the same reference Solver checkpoint at ∼8K cumulative training data. Values with arrows denote absolute point changes of Generator-guided training relative to Teacher-guided training at the same cumulative data scale.

| Training Method | Benchmark Datasets | | | | | | | Avg. |
|---|---|---|---|---|---|---|---|---|
| | AMC | Minerva | MATH-500 | GSM8K | Olympiad | AIME-25 | AIME-24 | |
| *∼8K cumulative data* | | | | | | | | |
| Reference Solver | 63.7 | 52.4 | 81.2 | 87.3 | 55.1 | 24.6 | 28.4 | 56.1 |
| *∼16K cumulative data* | | | | | | | | |
| Teacher-guided continuation | 67.2 | 55.1 | 84.3 | 88.9 | 58.4 | 27.8 | 32.5 | 60.3 |
| Generator-guided continuation (Ours) | 66.9 ↓0.3 | 55.8 ↑0.7 | 84.1 ↓0.2 | 89.2 ↑0.3 | 58.9 ↑0.5 | 28.3 ↑0.5 | 33.2 ↑0.7 | 60.5 ↑0.2 |
| *∼28K cumulative data* | | | | | | | | |
| Teacher-guided continuation | 68.9 | 57.3 | 85.6 | 89.6 | 60.7 | 31.0 | 43.0 | 62.3 |
| Generator-guided continuation (Ours) | 70.4 ↑1.5 | 59.2 ↑1.9 | 86.9 ↑1.3 | 90.1 ↑0.5 | 62.6 ↑1.9 | 33.3 ↑2.3 | 46.2 ↑3.2 | 64.1 ↑1.8 |
| *∼47K cumulative data* | | | | | | | | |
| Teacher-guided continuation | 69.6 | 58.1 | 86.0 | 89.9 | 61.2 | 31.8 | 43.0 | 62.8 |
| Generator-guided continuation (Ours) | 73.8 ↑4.2 | 62.6 ↑4.5 | 89.4 ↑3.4 | 91.0 ↑1.1 | 66.3 ↑5.1 | 38.1 ↑6.3 | 51.3 ↑8.3 | 67.5 ↑4.7 |

Table 6: Ablation of reward functions (averaged across benchmarks). $\rho$ represents the Solver success rate, $\mu$ the target success rate, and $\sigma$ the standard deviation. Values with arrows represent changes relative to our Gaussian baseline.

| REWARD FUNCTION | VALID (%) | AVG (%) | Δ (%) |
|---|---|---|---|
| $\mathcal{N}(\mu = 0.5, \sigma = 0.2)$ (OURS) | **89.9** | **35.72** | - |
| $\Psi_\rho(a = 0, b = 1)$ (UNIFORM) | 89.4 | 35.52 | ↓0.20 |
| $\Psi_\rho(a = 1, b = 0)$ (LINEAR) | 89.8 | 35.47 | ↓0.25 |
| $\Psi_\rho(a = -1, b = 1)$ (INVERSE) | 88.9 | 35.42 | ↓0.30 |
| $\mathcal{N}(\mu = 0.3, \sigma = 0.2)$ | 89.5 | 35.32 | ↓0.40 |
| $\mathcal{N}(\mu = 0.4, \sigma = 0.2)$ | 89.7 | 35.37 | ↓0.35 |
| $\mathcal{N}(\mu = 0.6, \sigma = 0.2)$ | 89.7 | 35.50 | ↓0.22 |
| $\mathcal{N}(\mu = 0.7, \sigma = 0.2)$ | 89.8 | 35.43 | ↓0.29 |

Table 7: Scaling analysis of Socratic-Zero. (a) Effect of Teacher scale with a fixed Solver (Qwen3-8B-Base), where improvements are measured relative to the same SFT baseline (35.9 average accuracy). (b) Solver performance under increasing cumulative curriculum data scales for Teacher-guided and Generator-guided continuation.

(a) Effect of Teacher Scale

| Teacher | Params | Scale | Avg. | Δ |
|---|---|---|---|---|
| Qwen3-4B-Base | 4B | 0.5× | 45.7 | +9.8 |
| Qwen3-8B-Base | 8B | 1.0× | 46.6 | +10.7 |
| Qwen3-14B | 14B | 1.75× | 49.9 | +14.0 |
| Qwen3-32B | 32B | 4.0× | 51.2 | +15.3 |
| Qwen3-235B-A22B | 235B | 29.4× | 56.1 | +20.2 |

(b) Data Scaling and Convergence

| Data | Teacher | Generator | Gap |
|---|---|---|---|
| ∼8K | 56.1 | 56.1 | – |
| ∼16K | 60.3 | 60.5 | +0.2 |
| ∼28K | 62.3 | 64.1 | +1.8 |
| ∼47K | 62.8 | 67.5 | +4.7 |

### 4.3 ABLATION AND SCALING ANALYSIS

**Reward Function Ablation.** Table 6 compares different reward functions used to weight Generator training examples. Our Gaussian reward, $\mathcal{N}(\mu = 0.5, \sigma = 0.2)$, achieves the best average performance among all tested alternatives. Uniform, linear, and inverse weighting functions all perform slightly worse, and shifting the Gaussian center away from $\mu = 0.5$ also leads to small but consistent degradation. This result further supports our design choice of emphasizing problems near the Solver's current capability boundary.

**Effect of Teacher Scale.** Larger Teachers improve final Solver performance, but smaller models already help: with the same 35.9 SFT baseline, 4B and 8B Teachers reach 45.7 and 46.6, while 14B, 32B, and 235B-A22B reach 49.9, 51.2, and 56.1. Thus Teacher scaling helps, yet improvement remains possible with a Teacher no larger than the Solver.

**Data Scaling and Convergence.** Table 7b summarizes Solver performance under increasing cumulative curriculum data scales. Both Teacher-guided and Generator-guided continuation improve over the same reference Solver as more data is added, but their scaling behavior differs. Teacher-guided continuation rises from 56.1 to 60.3 and 62.3, then slows to 62.8 at $\sim$47K cumulative data, whereas Generator-guided continuation reaches 60.5, 64.1, and 67.5 at the corresponding scales. As a result, the advantage of Generator-guided training grows from +0.2 to +4.7 points, indicating stronger sustained improvement at larger data scales.

**Interpretation of the Co-evolution Results.** The scaling results clarify how the three-agent loop contributes to the final Solver gains. Teacher scale affects the quality of the local curriculum signal, but the improvement is already visible with a Teacher no larger than the Solver. Once the Generator has learned this signal, continued training becomes less dependent on repeated calls to the largest Teacher: Generator-guided continuation improves from 60.5% at approximately 16K cumulative examples to 67.5% at approximately 47K, whereas Teacher-guided continuation reaches 62.8%. This pattern is consistent with the intended division of labor. The Teacher supplies verification and targeted refinements, the Solver exposes its current failure boundary, and the Generator turns those refinements into a reusable source of future problems. The result is a curriculum that changes with the Solver rather than a fixed augmentation set. These findings motivate the data accounting used throughout the paper: a one-time 1,500-pair cold-start SFT, a separate 100-question seed set, and approximately 7.9K co-evolution examples for the main Stage-3 result.

**Practical Implications.** The results also identify the operating conditions that make the loop useful in practice. The correctness signal must be cheap enough to apply to every Solver attempt, because the Teacher verifies both successful and failed trajectories before they enter the next stage. The curriculum should remain bounded by the Solver's observed failure distribution; otherwise, a Generator can amplify patterns that are novel but poorly matched to the learner. Our data accounting separates the cold-start supervision from the subsequent co-evolution examples so that improvements can be attributed to the adaptive loop rather than to an unreported increase in external data. The same separation is needed when comparing against prior self-improvement systems, whose data scales and verifier configurations differ. In future evaluations, we will report independent verifier checks, seed and filtering counts, and uncertainty across student runs. These measurements will make it possible to distinguish genuine curriculum improvement from changes in sampling, model scale, or evaluation protocol.

## 5 Conclusion and Future Work

In this paper, we introduced Socratic-Zero, a multi-agent co-evolutionary framework where Solver, Teacher, and Generator agents bootstrap autonomous mathematical reasoning from minimal seed data. Our implementation demonstrates that a carefully designed learning mechanism can achieve strong results without relying on massive external datasets, offering a practical path for reasoning systems in resource-constrained scenarios. The Solver supplies failure cases that define the next curriculum, keeping generated examples tied to the model's current capability boundary, while value-weighted Generator training amortizes the Teacher's problem-design strategy as training continues. In the controlled mathematical experiments, the seven-task average rises from 35.9 after cold-start SFT to 56.1 at the main Stage-3 point; the Generator reaches 36.04% downstream utility and 39.61% $\pm$ 0.17 in the expanded 10K evaluation. The strongest claims concern objectively verifiable mathematical tasks and the post-SFT bootstrapping loop; cross-domain results are secondary. The Teacher remains part of the verification path, so the method trades curated data for model inference and reliable correctness checks. The agent dynamics also lack a formal convergence analysis. Future work will study independent verifiers, calibrated confidence, and curriculum stability under noisy feedback before extending the framework to scientific discovery, real-world decision making, and other subjective domains. These boundaries define the next stage of the work: broader domains require stronger validation before autonomous expansion.

## REFERENCES


Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.13649.

Jiaqi Chen, Bang Zhang, Ruotian Ma, Peisong Wang, Xiaodan Liang, Zhaopeng Tu, Xiaolong Li, and Kwan-Yee K. Wong. SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning. arXiv preprint arXiv:2504.19162, 2025a. URL https://arxiv.org/abs/2504.19162.

Yue Chen, Minghua He, Fangkai Yang, Pu Zhao, Lu Wang, Yu Kang, Yifei Dong, Yuefeng Zhan, Hao Sun, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. WarriorMath: Enhancing the Mathematical Ability of Large Language Models with a Defect-aware Framework. arXiv preprint arXiv:2508.01245, 2025b. URL https://arxiv.org/abs/2508.01245.

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. URL https://arxiv.org/abs/2401.01335.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. arXiv preprint arXiv:2103.03874, 2021. URL https://arxiv.org/abs/2103.03874.

Chenghua Huang, Zhizhen Fan, Lu Wang, Fangkai Yang, Pu Zhao, Zeqi Lin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Self-Evolved Reward Learning for LLMs. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2411.00418.

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. URL https://arxiv.org/abs/2508.05004.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards Streamlined Distillation for Large Language Models. arXiv preprint arXiv:2402.03898, 2024. URL https://arxiv.org/abs/2402.03898.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. LLM2LLM: Boosting LLMs with Novel Iterative Data Enhancement. arXiv preprint arXiv:2403.15042, 2024. URL https://arxiv.org/abs/2403.15042.

Xuyang Liu, Zichen Wen, Shaobo Wang, Junjie Chen, Zhishan Tao, Yubo Wang, Tailai Chen, Xiangqi Jin, Chang Zou, Yiyu Wang, Chenfei Liao, Xu Zheng, Honggang Chen, Weijia Li, Xuming Hu, Conghui He, and Linfeng Zhang. Shifting AI Efficiency From Model-Centric to Data-Centric Compression. arXiv preprint arXiv:2505.19147, 2025. URL https://arxiv.org/abs/2505.19147.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. arXiv preprint arXiv:2303.17651, 2023. URL https://arxiv.org/abs/2303.17651.

Aru Maekawa, Satoshi Kosugi, Kotaro Funakoshi, and Manabu Okumura. DiLM: Distilling Dataset into Language Model for Text-level Dataset Distillation. arXiv preprint arXiv:2404.00264, 2024. URL https://arxiv.org/abs/2404.00264.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive Learning from Complex Explanation Traces of GPT-4. arXiv preprint arXiv:2306.02707, 2023. URL https://arxiv.org/abs/2306.02707.

Subhojyoti Mukherjee, Viet Dac Lai, Raghavendra Addanki, Ryan Rossi, Seunghyun Yoon, Trung Bui, Anup Rao, Jayakumar Subramanian, and Branislav Kveton. Offline RL by Reward-Weighted Fine-Tuning for Conversation Optimization. arXiv preprint arXiv:2506.06964, 2025. URL https://arxiv.org/abs/2506.06964.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv preprint arXiv:2305.18290, 2023. URL https://arxiv.org/abs/2305.18290.

Noveen Sachdeva and Julian McAuley. Data Distillation: A Survey. arXiv preprint arXiv:2301.04272, 2023. URL https://arxiv.org/abs/2301.04272.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. arXiv preprint arXiv:2009.01325, 2020. URL https://arxiv.org/abs/2009.01325.

Shaobo Wang, Yicun Yang, Zhiyuan Liu, Chenghao Sun, Xuming Hu, Conghui He, and Linfeng Zhang. Dataset distillation with neural characteristic function: A minmax perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25570–25580, June 2025. URL https://arxiv.org/abs/2502.20653.

Zeqiu Xu, Jiani Wang, Xiaochuan Xu, Peiyang Yu, Tianyi Huang, and Jingyuan Yi. A survey of reinforcement learning-driven knowledge distillation: Techniques, challenges, and applications. *Preprints.org*, 2025. doi: 10.20944/preprints202503.0903.v2. URL https://doi.org/10.20944/preprints202503.0903.v2.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Shaoxiong Zhan, Yanlin Lai, Ziyu Lu, Dahua Lin, Ziqing Yang, and Fei Tan. MathSmith: Towards Extremely Hard Mathematical Reasoning by Forging Synthetic Problems with a Reinforced Policy. arXiv preprint arXiv:2508.05592, 2025. URL https://arxiv.org/abs/2508.05592.

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. URL https://arxiv.org/abs/2505.03335.

# A Additional Ablation Studies

## A.1 Solver Optimization Objective

We compare DPO and GRPO for Solver optimization in the first stage of training. As shown in Table 8, DPO achieves better final Solver performance while also being more computationally efficient as an off-policy objective.

Table 8: Comparison of Solver optimization objectives in the first training stage.

| **Method** | **Solver Avg. Score (%)** |
|---|---|
| DPO | 38.7 |
| GRPO | 36.9 |

### A.2 Effect of Teacher Solution Injection

We further study the effect of teacher solution injection during training. The proportion of cases in which the Solver fully relies on the Teacher solution (i.e., all Solver attempts are incorrect) is 10%, 13%, and 14% across the three training stages, respectively, indicating that direct Teacher fallback remains limited throughout training.

Table 9 reports an ablation at the 8B scale, where the Teacher and Solver use the same base model family. Removing teacher solution injection reduces final performance from 46.6 to 45.4, but the framework still delivers a substantial improvement over the SFT baseline. This suggests that teacher solution injection is helpful but not the primary source of performance gains.

Table 9: Ablation on teacher solution injection at the 8B scale. Improvements are measured relative to the same SFT baseline.

| **Setting** | **Avg. (%)** | $\Delta$ |
|---|---|---|
| w/o teacher solution injection | 45.4 | +10.2 |
| w/ teacher solution injection | 46.6 | +11.4 |

### A.3 Verifier Configuration

To examine sensitivity to verifier choice, we fix Qwen3-235B-A22B as the problem enhancer and compare four verifier configurations: (1) Qwen3-235B as the original verifier, (2) MathRule only as a purely rule-based verifier, (3) Qwen3-32B as a smaller LLM verifier, and (4) GPT-OSS-120B as an alternative verifier from a different model family. All other settings remain unchanged.

Table 10 shows that replacing the original verifier with other LLM-based verifiers leads to only minor performance changes, whereas relying on a purely rule-based verifier results in a noticeably larger drop. This indicates that the framework is robust to the specific verifier model, provided that the verifier retains sufficient semantic evaluation capability.

Table 10: Ablation on verifier configuration.

| **Verifier Configuration** | **Final Avg. (%)** | **$\Delta$ vs. Original** |
|---|---|---|
| Qwen3-235B (Original) | 56.1 | Baseline |
| MathRule Only | 51.3 | $-4.8$ |
| Qwen3-32B | 55.7 | $-0.4$ |
| GPT-OSS-120B | 55.9 | $-0.2$ |

### A.4 Cost-Performance Comparison

We further compare Socratic-Zero against prior self-improvement and data-synthesis methods in terms of both training cost and final performance. Since most related works do not explicitly report exact GPU hours or precise generated data volumes, the numbers in Table 11 are approximate estimates based on their reported training setups, data scales, and model configurations. For Socratic-Zero, we additionally convert Teacher inference cost into equivalent A100 GPU hours and include it in the total cost.

Table 11 shows that Socratic-Zero achieves a favorable cost–performance trade-off across multiple configurations. Even with an 8B Teacher throughout the self-play pipeline, our method remains competitive while using relatively modest total compute. With a stronger 235B Teacher and a distilled 32B Generator, Socratic-Zero reaches the best overall performance in our comparison. These results suggest that the framework remains effective under lightweight settings, while also scaling well when paired with stronger external Teachers and a larger Generator.

Table 11: Approximate cost–performance comparison with prior self-improvement and data-synthesis methods. Total cost is reported in equivalent A100 GPU hours. For our method, Teacher inference cost is included in the estimate. Gains are measured relative to the same SFT baseline.

| Method | Models Trained | Generated Data | Peak Memory | Total Cost | Avg. | Gain |
|---|---|---|---|---|---|---|
| Absolute Zero | 1×8B (joint) | ∼40K | 8×96GB | ∼384 A100h | 52.6 | +16.7 |
| SPIRAL | 2×8B | ∼51K | 8×96GB | ∼672 A100h | 47.0 | +4.4 |
| R-Zero | 2×8B | ∼24K | 8×96GB | ∼576 A100h | 54.7 | +18.8 |
| SPICE | 1×8B (joint) | ∼82K (20K docs) | 8×96GB | >1000 A100h | 53.8 | +17.9 |
| Ours (8B Teacher, self-play) | 1×8B (Solver) | ∼4.8K | 8×96GB | ∼98 A100h | 47.6 | +5.0 |
| Ours (235B Teacher) | 1×8B (Solver) | ∼5.2K | 8×96GB | ∼112 A100h | 56.1 | +13.5 |
| Ours (8B Teacher, 8B Generator) | 1×8B (Solver) + 1×8B (Generator) | ∼16K | 8×96GB | ∼382 A100h | 59.3 | +16.7 |
| Ours (235B Teacher, 32B Generator) | 1×8B (Solver) + 1×32B (Generator) | ∼18K | 8×96GB | ∼492 A100h | 67.5 | +24.9 |

# B PROMPTS

## B.1 SOLVER REASONING PROMPT

**Solver Mathematical Reasoning Prompt**

You are an IMO gold medalist solving a computational math competition problem.
Understand: Restate the problem mathematically. Identify knowns, unknowns, and constraints.
Plan: Choose an efficient method, show clear logic.
Execute: Show all key steps – algebra, number theory, or combinatorics. No skipped calculations.
Verify: Check with small cases, reverse substitution, or estimation.
Conclude with the exact answer in LaTeX: \[\boxed{< answer >}\]
Given Problem: {question}

## B.2 TEACHER EVALUATION PROMPT

**Teacher Solution Grading Prompt**

You are a professional math teacher responsible for grading and error analysis.
Grading criteria: Focus on final answer correctness, use reference when provided, provide concise error analysis for incorrect answers.
Return JSON format:

```
{
  "correct_answers": ["correct answer 1", "correct answer 2"],
  "incorrect_answers": [
    {"answer": "incorrect answer", "analysis": "brief error analysis"}
  ]
}
```

Problem: {question} | Reference: {reference_info} | Student answers: {student_answers}

## B.3 TEACHER GENERATION PROMPT

**Teacher Problem Enhancement Prompt**

You are a math problem enhancement expert specializing in competition-style mathematics. Generate enhanced problems based on student error analysis with complete solutions.
Requirements: Generate enhanced problem, provide detailed solution, ensure solvability and correctness.
Enhancement principles: Target specific error points, maintain mathematical essence, help avoid similar errors.
Return JSON format:

```
{
  "enhanced_question": "enhanced problem content",
  "solution": "detailed solution steps",
  "answer": "final answer"
}
```

Original: {original_question} | Error analysis: {error_analysis}

### B.4 Static Augmentation Baseline Prompts

**Static Augmentation Evolution Prompts**

**Upward Evolution:** Step 1: Identify elements that can increase complexity. Step 2: Plan to modify at least three components. Step 3: Implement rewritten instruction. Step 4: Review and provide final version.
**Downward Evolution:** Step 1: Identify elements that can decrease complexity. Step 2: Plan to simplify at least three components. Step 3: Implement easier version. Step 4: Review and provide final simplified version.
Format: Step 1 #Elements#: | Step 2 #Plan#: | Step 3 #Rewritten#: | Step 4 #Final#:

## C Implementation Details

We conducted training experiments on 8ŒNVIDIA H20 GPUs with the following configuration: - GPU Memory: 96GB HBM3 per GPU - Total Training Memory: 768GB - Interconnect: NVLink 4.0 - Storage: High-speed NVMe SSD arrays for dataset caching - Network: InfiniBand for distributed training coordination

The training infrastructure utilized mixed-precision training (FP16) with gradient checkpointing to optimize memory usage. We employed distributed training using PyTorch's DistributedDataParallel with NCCL backend for efficient gradient synchronization across GPUs.

### C.1 Training Hyperparameters

We provide the complete hyperparameter settings for all components of the Socratic-Zero framework in Table 12.

## D Curriculum Evolution Details

This appendix provides detailed mechanisms for curriculum evolution and problem categorization that were omitted from the main methodology for brevity.

### D.1 Problem Categorization Strategy

For each problem $q \in \mathcal{C}_t$, the Solver generates $k$ solution attempts, yielding a success count $z_q = \sum_{i=1}^{k} \mathcal{T}_{\text{eval}}(q, a_i)$. This diagnostic metric enables the Teacher to categorize problems into three distinct zones based on the Solver's current capability:

1. **Mastered Zone** ($\mathcal{C}_{\text{mastered}} = \{q \mid z_q = k\}$): Problems that the Solver consistently solves correctly across all $k$ attempts. These problems serve as positive reinforcement and provide a foundation for generating slightly more challenging variants.
2. **Learning Zone** ($\mathcal{C}_{\text{learning}} = \{q \mid 0 < z_q < k\}$): Problems that the Solver solves partially–some attempts succeed while others fail. This zone represents the optimal frontier for learning, where the Solver demonstrates emerging competence but has not yet achieved mastery.
3. **Too Difficult Zone** ($\mathcal{C}_{\text{difficult}} = \{q \mid z_q = 0\}$): Problems that consistently result in failure across all attempts. These problems are temporarily deferred to prevent the Solver from being overwhelmed by tasks beyond its current capability.

### D.2 Adaptive Problem Generation

New problems are strategically generated only from the *Mastered* and *Learning Zone* categories, ensuring that curriculum expansion remains within the Solver's zone of proximal development. The generation process follows:

$$\Delta\mathcal{C}_t = \{\mathcal{T}_{\text{gen}}(q, a_S) \mid q \in \mathcal{C}_{\text{learning}} \cup \mathcal{C}_{\text{mastered}},\ a_S \text{ is a failed attempt on } q\}\,. \tag{11}$$

This strategy ensures that:

Table 12: Hyperparameters used in Socratic-Zero framework.

| Component | Parameter | Value |
|---|---|---|
| Solver SFT Training | Learning rate | 5e-5 |
| | Per-device batch size | 2 |
| | Gradient accumulation steps | 4 |
| | Maximum sequence length | 2048 |
| | LoRA rank ($r$) | 64 |
| | LoRA alpha ($\alpha$) | 128 |
| | LoRA dropout | 0.1 |
| | Number of epochs | 1 |
| Solver DPO Training | Learning rate | 1e-6 – 5e-6 |
| | Per-device batch size | 2 |
| | Gradient accumulation steps | 4 – 16 |
| | Maximum sequence length | 2048 |
| | Maximum training steps | 10 – 200 |
| | DPO regularization ($\beta$) | 0.05 – 0.2 |
| | Warmup steps | 2 – 20 |
| | Optimizer | AdaFactor |
| | Weight decay | 0.01 |
| | Maximum gradient norm | 1.0 |
| Generator Training | Learning rate | 1e-5 |
| | Per-device batch size | 1 |
| | Gradient accumulation steps | 8 |
| | Maximum sequence length | 2048 |
| | LoRA rank ($r$) | 64 |
| | LoRA alpha ($\alpha$) | 128 |
| | Number of epochs | 2 |
| Curriculum Parameters | Solutions per problem ($k$) | 8 |
| | reward mean ($\mu$) | 0.5 |
| | reward std ($\sigma$) | 0.2 |
| | Historical replay ratio | 25% |
| Evaluation Settings | Sampling temperature | 0.7 |
| | Number of samples | 32 |
| | Token limit for validity check | 4096 |

- **From Mastered Problems**: The Teacher generates slightly more challenging variants, pushing the boundary of the Solver's competence.
- **From Learning Zone Problems**: The Teacher creates problems that address specific failure modes, helping the Solver overcome particular reasoning gaps.
- **Exclusion of Too Difficult Problems**: Problems that consistently fail are not used for generation, preventing the creation of even more challenging problems that would be counterproductive.

### D.3 Dynamic Recategorization

As the Solver evolves through training, problems are dynamically recategorized based on updated performance:

$$\mathcal{C}^{(t)}_{\text{difficult}} \to \mathcal{C}^{(t+1)}_{\text{learning}} \quad \text{(capability improvement)} \tag{12}$$

$$\mathcal{C}^{(t)}_{\text{learning}} \to \mathcal{C}^{(t+1)}_{\text{mastered}} \quad \text{(skill consolidation)} \tag{13}$$

$$\mathcal{C}^{(t)}_{\text{mastered}} \to \mathcal{C}^{(t+1)}_{\text{learning}} \quad \text{(rare: capability regression)} \tag{14}$$

This dynamic recategorization ensures that the curriculum remains responsive to the Solver's evolving capabilities, automatically adjusting the difficulty distribution as learning progresses.

# E TEACHER MODEL INFRASTRUCTURE

The Teacher model (Qwen3-235B-A22B-Instruct-2507) requires substantial computational resources for curriculum generation and solution evaluation. We deployed the model using a distributed inference architecture to meet the throughput demands of the co-evolutionary training process.

We distributed the Teacher model across 16 AMD MI308X GPUs, each equipped with 192GB HBM3 memory, providing a total of 3,072GB aggregate memory. This configuration enables concurrent processing of curriculum generation requests while maintaining inference consistency across the framework.

To ensure system reliability and scalability, we implemented a multi-endpoint architecture with automatic load balancing and failover mechanisms. We configured the inference service with connection pooling (50 concurrent connections per endpoint) and exponential backoff retry policies to handle high request volumes during training.

We optimized key performance parameters for the mathematical reasoning domain: request timeouts of 600 seconds accommodate complex problem generation, while a 4,096-token limit ensures efficient solution evaluation. Batch processing utilizes 32 concurrent workers to maximize throughput during curriculum evolution phases.

# F TEACHER-GENERATED PROBLEM ENHANCEMENT

We provide examples of how the Teacher model enhances problems based on Solver failures. The following demonstrates the progression from original problems to highly-targeted enhanced versions.

## F.1 EXAMPLE 1: RATIONAL INEQUALITY ENHANCEMENT

**Original Problem:**

Find all real numbers $x$ satisfying $\frac{2x-5}{x+3} \geq 2$. (Give your answer in interval notation.)

**Enhanced Problem (Round 3):**

Find all real numbers $x$ satisfying $\frac{2x-5}{x^2-9} + \frac{1}{x+3} \leq \frac{4x+1}{(x-3)^2}$. (Give your answer in interval notation.)

**Enhancement Analysis:** The enhancement introduces multiple complexity factors: (1) factored denominators requiring domain analysis, (2) multiple rational terms requiring common denominators, (3) squared terms in denominators, and (4) more complex algebraic manipulation. The enhanced problem targets common student errors in rational inequality solving while maintaining the core mathematical concepts.

## F.2 EXAMPLE 2: NUMBER THEORY ENHANCEMENT

**Original Problem:**

Find the greatest common divisor of $10! + 6$ and $11! + 14$.

**Enhanced Problem:**

Find the greatest common divisor of $12! + 8$ and $13! + 26$, where the second number can be written as $13 \cdot 12! + 26$.

**Enhancement Analysis:** The enhancement maintains the GCD structure while increasing numerical complexity and requiring students to recognize the relationship between consecutive factorials, targeting errors in modular arithmetic applications.

Table 13: Seed Problem Distribution Across Mathematical Domains

| Subject Area | Count | Representative Topics |
|---|---|---|
| *Algebra* | 15 | Linear/quadratic equations, inequalities, functions |
| *Number Theory* | 15 | Divisibility, modular arithmetic, prime factorization |
| *Geometry* | 15 | Coordinate geometry, trigonometry, area/volume calculations |
| *Combinatorics* | 15 | Counting principles, permutations, probability |
| *Precalculus* | 15 | Complex numbers, sequences, polynomial analysis |
| *Intermediate Algebra* | 15 | Advanced algebraic manipulation, systems |
| *Prealgebra* | 10 | Foundational arithmetic and basic algebraic concepts |
| **Total** | **100** | **Comprehensive mathematical reasoning coverage** |

## G SEED SELECTION PROTOCOL

The selection of initial seed problems is critical for establishing an effective curriculum foundation. We employed a systematic approach to ensure the seed set provides appropriate difficulty, comprehensive coverage, and sufficient diversity for subsequent curriculum evolution.

**Difficulty Alignment** We selected seed problems to match the base model's capability range to ensure productive learning dynamics. We drew problems from MATH dataset Levels 2-4, which empirically provide optimal challenge levels for our base models. Specifically, we excluded Level 1 problems (too easy, leading to trivial curriculum generation) and Level 5 problems (too difficult, resulting in universal failure and poor learning signals). Pre-filtering involved evaluating candidate problems with the base model using 8 solution attempts; we retained problems with success rates between 10-70% to ensure neither complete failure nor trivial success.

**Domain Coverage** To ensure comprehensive mathematical reasoning development, we sampled seed problems across all seven MATH subject areas with balanced representation as shown in Table 13:

This distribution ensures that curriculum evolution can target weaknesses across diverse mathematical domains rather than overfitting to specific problem types.

**Diversity Assurance** Within each subject area, we selected problems to maximize methodological diversity. We employed clustering based on solution approach similarity (using embedding representations of problem statements) and selected problems from different clusters to ensure varied reasoning patterns. Additionally, we explicitly included problems requiring different mathematical tools to promote comprehensive skill development.

**Quality Control** We subjected all candidate problems to rigorous quality verification through a multi-stage process:

**Quality Control Pipeline**

1. *Clarity Check*: Problems must have unambiguous statements and well-defined solution paths
2. *Answer Verification*: We validated reference solutions by the Teacher model with multiple independent attempts
3. *Value*: Problems must demonstrate clear learning objectives and avoid trick questions or overly specialized knowledge
4. *Contamination Avoidance*: We excluded seed problems from all evaluation benchmarks to prevent data leakage

This systematic selection process ensures that the initial curriculum $\mathcal{C}_0$ provides a robust foundation for the co-evolutionary training dynamics while maintaining the diversity necessary for effective curriculum expansion.

## H Evaluation Protocol Details

**Mean@32 Sampling Strategy** The Mean@32 evaluation metric represents the average accuracy across 32 independent solution attempts per problem. For each test problem, we generated 32 distinct solutions using temperature-based sampling (T=0.7) with top-p nucleus sampling (p=0.9) as specified in Table 12. This approach provides robust performance estimates by capturing the model's consistency and reliability across multiple attempts.

We employed the sampling process using zero-shot prompting without few-shot examples to ensure unbiased evaluation. We generated each of the 32 solutions independently with different random seeds, preventing potential correlation effects. The final accuracy is computed as the proportion of correct solutions among the 32 attempts, providing a more stable performance measure than single-shot evaluation.

**MathRule Answer Extraction** MathRule is a rule-based tool designed to extract and standardize final numerical answers from mathematical solution text. The tool employs pattern matching to identify answer indicators such as "Therefore," "Thus," "The answer is," and LaTeX boxed expressions like \boxed{}.

The extraction process involves: (1) Locating answer indicators within the solution text, (2) Parsing mathematical expressions using regex patterns for common formats (fractions, decimals, integers, algebraic expressions), (3) Standardizing representations (e.g., converting $\frac{1}{2}$ to 0.5 when appropriate), (4) Handling multiple answer formats and selecting the most confident extraction based on contextual cues.

MathRule achieves high precision in answer extraction while maintaining robustness to variations in solution formatting and mathematical notation styles.

**LLM Judge Configuration** The Teacher model (Qwen3-235B-A22B-Instruct-2507) serves as an LLM judge for semantic validation when rule-based extraction is insufficient or ambiguous. The judge evaluates both numerical correctness and reasoning validity using structured prompts.

We instructed the evaluation prompt to: (1) Verify the final numerical answer against the expected result, (2) Assess the logical coherence of the reasoning steps, (3) Identify any mathematical errors or invalid assumptions, (4) Provide binary correctness judgments with brief justification.

We ensured judge reliability through temperature 0.1 sampling for consistent evaluations and validation against human expert annotations on a subset of problems. The dual-verification approach (MathRule + LLM judge) provides reliable automated assessment for large-scale evaluation.

## I Problem Quality Control Mechanism

To ensure curriculum integrity and prevent the propagation of erroneous problems, we implemented a comprehensive quality control mechanism that monitors problem validity through Solver performance feedback and automated verification.

Table 14: Solver Mean Reward Evolution Across Training Stages

| Stage | S1 | S2 | S3 | Trend |
|---|---|---|---|---|
| Mean Reward (%) | 52.1 | 48.7 | 50.1 | $\downarrow$ then $\uparrow$ |

**Teacher Self-Verification Protocol** When the Teacher model evaluates Solver attempts and finds that all $k = 8$ solutions for a given problem are incorrect (success rate $j_p = 0$), this triggers an automatic quality verification process. The system recognizes that universal failure may indicate either: (1) the problem exceeds current Solver capability (expected behavior), or (2) the problem itself or its reference solution contains errors (quality issue).

The Teacher model performs self-verification by re-examining both the problem statement and its originally provided reference solution. This involves: (1) Re-solving the problem independently with temperature 0.1 for consistency, (2) Cross-validating the reference solution against the new solution attempt, (3) Checking for mathematical consistency, ambiguous problem statements, or computational errors, (4) Verifying that the problem has a unique, well-defined solution.

**Problem Filtering and Exclusion** We immediately flagged and excluded problems that fail the self-verification process from further curriculum evolution. Specifically, we discarded problems if: (1) The Teacher cannot reproduce its own reference solution, (2) Multiple valid interpretations of the problem statement exist, (3) Computational errors are detected in the reference solution, (4) The problem lacks sufficient information for a unique solution.

We removed discarded problems from the active curriculum $\mathcal{C}_t$ and they do not contribute to subsequent Solver training or Generator learning. This prevents the accumulation of low-quality problems that could degrade training effectiveness or introduce systematic biases.

**MathRule Integration for Contamination Minimization** The integration of MathRule answer extraction serves as an additional quality control layer by providing objective, rule-based verification independent of LLM judgment. When MathRule successfully extracts a clear numerical answer from the Solver's solution, this extraction is compared against the reference answer using standardized formats.

This dual-verification approach (MathRule + Teacher evaluation) minimizes contamination from: (1) LLM judge inconsistencies or biases, (2) Format-related misinterpretations, (3) Numerical precision issues, (4) Ambiguous answer representations.

Problems where MathRule and Teacher evaluations consistently disagree trigger additional quality review, as such disagreements often indicate underlying issues with problem clarity or reference solution accuracy.

**Feedback-Driven Quality Monitoring** The system continuously monitors curriculum quality through Solver performance patterns. We flagged problems that consistently produce anomalous results–such as sudden performance drops across multiple Solver variants or inconsistent difficulty ratings–for manual review or automatic exclusion.

This feedback-driven approach ensures that quality control adapts to emerging issues and maintains curriculum integrity throughout the co-evolutionary training process, preventing the accumulation of problematic content that could compromise learning effectiveness.

## J Curriculum Stability and Diversity Analysis

We analyzed the curriculum evolution dynamics across two dimensions: difficulty progression and problem diversity preservation.

**Solver Performance Evolution** Table 14 tracks the Solver's mean reward (correctness rate) across training stages, revealing adaptive curriculum difficulty.

Table 15: Generator Reward Distribution Analysis

| Stage | S1 | S2 | S3 | Stability |
|---|---|---|---|---|
| High Reward Problems (%) | 50.7 | 49.4 | 50.2 | Stable |
| Target Range (45-55%) | ✓ | ✓ | ✓ | Maintained |

The Solver exhibits characteristic performance decline from Stage 1 (52.1%) to Stage 2 (48.7%) followed by recovery in Stage 3 (50.1%) as shown in Table 14. This pattern reflects adaptive curriculum generation where the Teacher progressively increases difficulty faster than Solver capability initially improves, then the Solver begins adapting to enhanced curriculum complexity.

**Generator Stability** Table 15 examines reward distribution in Generator training.

As demonstrated in Table 15, the Generator maintains remarkable stability with high-reward problems consistently around 50%, fluctuating within only 1.3% range. This indicates successful learning of the optimal difficulty zone defined by the Gaussian reward function with parameters $\mu = 0.5$ and $\sigma = 0.2$ as specified in Table 12.

**Problem Diversity** Three key mechanisms ensure curriculum diversity throughout training:

Multi-domain initialization: The 100 seed problems span all 7 MATH subjects (Algebra, Number Theory, Geometry, etc.) across difficulty levels 2-4 as detailed in Table 13, providing diverse starting points for curriculum evolution.

High-temperature sampling: We employed temperature 0.8-0.9 sampling at three critical stages: (1) Solver trajectory generation during curriculum advancement, (2) Teacher error analysis for varied failure interpretation, and (3) Teacher problem generation for diverse enhancement strategies.

Compounding diversity effects: Multi-domain seeds combined with stochastic sampling create diverse failure patterns, while high-temperature generation ensures varied problem formulations even from similar error patterns.

## K Generalizability of Problem Generation Capabilities

A key question emerging from our work is whether the Generator's learned problem creation abilities can transfer to domains beyond mathematical reasoning. The value function and curriculum evolution mechanisms developed in Socratic-Zero are domain-agnostic in principle, suggesting potential for broader applicability.

The Generator learns fundamental skills in difficulty calibration, error pattern recognition, and targeting that may generalize across reasoning domains. For instance, the ability to identify when a problem is "appropriately challenging" (around 50% success rate as shown in Table 15) represents a meta-cognitive skill applicable to logical reasoning, scientific problem-solving, or even creative tasks. The Gaussian reward function with $\mu = 0.5$ and $\sigma = 0.2$ (Table 12) creates a transferable framework for difficulty calibration that could adapt to other domains by adjusting the target success rate parameter.

Our corrected seven-task comparison gives the Socratic Generator 36.04% downstream utility (Table 4). In the expanded 10K-problem evaluation, it reaches 39.61% $\pm$ 0.17 versus 38.79% $\pm$ 0.23 for the strongest commercial generator. These results suggest that strategic specialization can outperform raw parameter scaling when the student-training protocol is held fixed.

However, domain transfer would require careful adaptation of the Teacher's evaluation capabilities and problem generation templates. Mathematical reasoning benefits from relatively objective correctness criteria with our dual-verification approach (MathRule + LLM judge) achieving 94.2% agreement with human experts, while other domains may require more nuanced evaluation frameworks. The seed selection protocol detailed in Table 13, which ensures balanced coverage across seven mathematical domains, provides a template for systematic domain expansion that could be adapted to physics, computer science, or other reasoning areas.

Future work should investigate whether a Generator trained on mathematical problems can effectively create challenging problems in adjacent domains like physics or computer science, potentially through few-shot adaptation or domain-specific fine-tuning leveraging the value learning mechanisms demonstrated in our framework.

## L Framework Scalability and Extensibility

The modular architecture of Socratic-Zero demonstrates strong potential for scalability and extension across multiple dimensions. The clear separation between Solver, Teacher, and Generator roles enables independent scaling and optimization of each component, as evidenced by our successful deployment across different computational configurations detailed in Table 12.

The framework's extensibility is particularly evident in its ability to accommodate different model architectures and scales. Our cross-model validation demonstrates consistent performance improvements: Qwen3-8B achieves 56.1% average accuracy (+20.2 points), while similar gains are observed on GLM4-9B and Qwen3-14B architectures. This cross-architecture consistency suggests the co-evolutionary principles transcend specific model families and could readily incorporate emerging architectures or specialized reasoning models.

The curriculum evolution mechanism shows robust scalability properties. Starting from just 100 seed problems (Table 13), the system generates thousands of highly valuable problems while maintaining quality, with our Generator achieving 95.6% validity rate compared to 89.1% from the base Qwen3-32B model (Table 3). This demonstrates that the framework can scale curriculum generation without proportional increases in seed data requirements.

Multi-domain extension represents another promising direction supported by our balanced seed distribution across seven mathematical domains. The current mathematical focus could expand to encompass multiple reasoning domains simultaneously, with domain-specific Teachers providing specialized curriculum generation while sharing the underlying co-evolutionary dynamics. The reward distribution analysis (Table 15) shows stable performance across training rounds, indicating the framework's robustness to curriculum expansion.

The framework also supports hierarchical scaling, where multiple Solver-Generator pairs could operate at different difficulty levels or specialization areas, coordinated by higher-level meta-learning mechanisms. The oscillatory convergence patterns observed in Table 14 suggest natural synchronization points where multiple agents could coordinate their learning phases.

## M Convergence and Theoretical Foundations

The theoretical understanding of multi-agent co-evolutionary learning remains an open challenge with significant implications for system reliability and predictability. Our empirical observations provide crucial insights into the convergence behavior of such systems.

The oscillatory convergence patterns documented in Table 14 reveal characteristic dynamics: Solver performance declines from R1 (60.12%) to R4 (48.7%) followed by recovery in R5 (50.1%). This pattern reflects adaptive curriculum generation where the Teacher progressively increases difficulty faster than Solver capability initially improves, then the Solver adapts to enhanced curriculum complexity. These bounded oscillations suggest the system reaches dynamic equilibria rather than static optima.

Complementing this, the Generator maintains remarkable stability with high-reward problems consistently around 50%, fluctuating within only 1.3% range across training rounds (Table 15). This stability indicates successful learning of the optimal difficulty zone defined by the Gaussian reward function with $\mu = 0.5$ and $\sigma = 0.2$ (Table 12), providing empirical evidence for convergence to meaningful equilibria.

The cross-architecture consistency observed in Table, where similar improvement patterns emerge across Qwen3-8B, GLM4-9B, and Qwen3-14B models, suggests robust system-level properties that transcend specific model architectures. This consistency provides evidence that the convergence behavior represents fundamental properties of the co-evolutionary dynamics rather than architecture-specific artifacts.

Future theoretical work should investigate conditions under which the system exhibits stable convergence versus chaotic dynamics. Key questions include: What curriculum evolution rates ensure stable learning? How do different value functions affect convergence properties? Can we establish bounds on the oscillation amplitude around target performance levels observed in our empirical data?

The intersection of curriculum learning, preference optimization, and multi-agent dynamics presents rich opportunities for theoretical development. The DPO training parameters (Table 12) and their interaction with curriculum evolution rates could inform theoretical models of multi-agent preference learning. Establishing convergence guarantees would enable more principled hyperparameter selection and provide confidence bounds for practical deployment.

# N Computational Cost Analysis

## N.1 Generator vs Teacher: Trainability and Deployment Cost

**Trainability Advantage:** A critical distinction between the Generator (Socratic-Generator-32B) and Teacher (Qwen3-235B-A22B-Instruct-2507) is **trainability**. The Generator can be directly fine-tuned and integrated into the co-evolution loop, enabling interactive curriculum adaptation based on Solver feedback. In contrast, the Teacher model is a frozen proprietary checkpoint that cannot be trained or modified, limiting its ability to adapt to evolving curriculum needs. This makes the Generator not only a cost-effective alternative but also a more flexible component for iterative improvement.

**Inference Deployment Cost:** We compare the relative computational requirements for deploying both models in curriculum generation:

- **Model Size**: Generator (32B parameters) vs Teacher (235B parameters) — **7.3× reduction**
- **Memory Footprint**: Accounting for KV cache, activations, and other overheads, Generator requires approximately **1/5-1/6** of the total GPU memory compared to Teacher
- **Inference Throughput**: Smaller models typically achieve higher tokens/second; Generator is estimated to be **3-5× faster**
- **Hardware Requirements**: Generator runs on the same standard GPU cluster as Solver training, while Teacher requires dedicated high-memory multi-GPU inference infrastructure

**Practical Implications:** By replacing the Teacher with the Generator, Socratic-Zero eliminates the need for separate high-end inference infrastructure. The entire system (Generator + Solver) can be deployed on a homogeneous GPU cluster, reducing hardware diversity, operational complexity, and total cost of ownership. Combined with the trainability advantage, this makes the Generator-guided approach significantly more practical for resource-constrained environments while maintaining near-identical performance (0.2pp gap in Table 5).

# O Detailed End-to-End Training Procedure

This appendix provides a comprehensive pseudocode description of the complete Socratic-Zero training process, integrating all key components discussed in Section 3: Solver improvement via DPO, curriculum enhancement by the Teacher, and Generator training via WSFT.

For the reported Qwen3-8B-Base experiments, $D_{\text{sft}}$ contains 1,500 original MATH Level-5 problem-solution pairs and is used once for cold-start instruction and output-format alignment. It is separate from the 100 curated seed problems in $C_0$, which initialize the subsequent co-evolution stage.

**Initialization (Lines 1-3):** The Solver is initialized through supervised fine-tuning on the 1,500-pair $D_{\text{sft}}$, while the Generator starts from its base checkpoint. The initial curriculum $C_0$ then consists of 100 carefully selected seed problems (detailed in Appendix G).

**Algorithm 1** Detailed End-to-End Training Procedure of Socratic-Zero

**Require:** Initial curriculum $C_0$, base Solver/Generator parameters $\theta_S^{\text{base}}/\theta_G^{\text{base}}$, fixed Teacher $T$ $(T_{\text{eval}}, T_{\text{gen}})$
**Require:** Total stages $T$, SFT data $D_{\text{sft}}$, attempts per problem $k$, DPO beta $\beta$, WSFT target success rate $\mu$ and std $\sigma$
1: $\theta_S^0 \leftarrow \text{SFT}(\theta_S^{\text{base}}, D_{\text{sft}})$ ▷ Initial fine-tuning of the Solver
2: $\theta_G^0 \leftarrow \theta_G^{\text{base}}$
3: $C_0 \leftarrow$ Select 100 seed problems ▷ details in Appendix G
4: **for** $t = 0$ to $T - 1$ **do**
5: **print**(f"— Starting Stage $t + 1$ —")
6: $D_{\text{pref}} \leftarrow \emptyset$, $F_t \leftarrow \emptyset$ ▷ Initialize preference and failure sets for this stage
7: **for** each problem $q \in C_t$ **do**
8: $\{a_i^S\}_{i=1}^k \leftarrow$ Sample $k$ solutions from $\pi_{\theta_S^t}(\cdot|q)$ ▷ Solver generates attempts
9: $Z^+, Z^- \leftarrow$ Partition$(\{a_i^S\})$ using $T_{\text{eval}}$ ▷ Teacher evaluates attempts (Eq. 4, 5)
10: $D_{\text{pref}} \leftarrow D_{\text{pref}} \cup \{(a^+, a^-) \mid a^+ \in Z^+, a^- \in Z^-\}$ ▷ Build preference pairs
11: $F_t \leftarrow F_t \cup \{(q, a^S) \mid a^S \in Z^-\}$ ▷ Collect failed attempts for enhancement
12: **end for**
13: $\pi_{\text{ref}} \leftarrow \pi_{\theta_S^t}$ ▷ Set reference policy for DPO
14: $\theta_S^{t+1} \leftarrow$ Update $\theta_S^t$ using DPO loss on $D_{\text{pref}}$ with $\pi_{\text{ref}}$ ▷ Solver update (Eq. 6)
15: $D_{\text{new_map}} \leftarrow \{(q, a^S) : T_{\text{gen}}(q, a^S) \mid (q, a^S) \in F_t\}$ ▷ Teacher enhances failed problems
16: $D_G \leftarrow \emptyset$ ▷ Initialize Generator training data
17: **for** each $((q, a^S), (q^T, a^T)) \in D_{\text{new_map}}$.items() **do**
18: $s_{q^T} \leftarrow$ Estimate success rate of $\pi_{\theta_S^{t+1}}$ on $q^T$ ▷ Evaluate new problem's utility
19: $v(q^T) \leftarrow \exp(-(s_{q^T}/k - \mu)^2/(2\sigma^2))$ ▷ Calculate value (Eq. 7)
20: $D_G \leftarrow D_G \cup \{(\text{input} = q, \text{target} = q^T, \text{weight} = v(q^T))\}$ ▷ Build weighted training tuple
21: **end for**
22: $\theta_G^{t+1} \leftarrow$ Update $\theta_G^t$ using WSFT loss on $D_G$ ▷ Generator distillation (Eq. 8)
23: $C_{t+1} \leftarrow C_t \cup \{q^T \mid (q^T, a^T) \in D_{\text{new_map}}.\text{values}()\}$ ▷ Augment curriculum for next stage
24: **end for**
25: **return** Final Solver parameters $\theta_S^T$ and Generator parameters $\theta_G^T$

**Phase 1 - Solver Evaluation (Lines 5-10):** For each problem in the current curriculum, the Solver generates $k$ solution attempts. The Teacher's evaluation component $T_{\text{eval}}$ partitions these attempts into correct ($Z^+$) and incorrect ($Z^-$) sets, forming preference pairs for DPO training. Failed attempts are collected in $F_t$ for subsequent curriculum enhancement.

**Phase 2 - Solver Update (Lines 11-12):** The Solver is updated using Direct Preference Optimization (DPO) on the collected preference pairs, with the current policy serving as the reference policy (Eq. 6).

**Phase 3 - Curriculum Enhancement & Generator Training (Lines 13-19):** The Teacher's generation component $T_{\text{gen}}$ enhances problems where the Solver failed. Each enhanced problem is assigned a utility value based on its estimated difficulty for the updated Solver (Eq. 7). The Generator learns to mimic the Teacher's enhancement strategy through Weighted Supervised Fine-Tuning (WSFT), where training samples are weighted by their utility values (Eq. 8).

**Phase 4 - Curriculum Update (Line 20):** The curriculum is augmented with newly generated problems, creating a progressively more challenging and diverse training set for the next stage.

This iterative process continues for $T$ stages, resulting in a co-evolved Solver and Generator that can autonomously improve without further Teacher guidance.

## P FROM-ZERO SEED ABLATION STUDY

To validate the robustness of our framework to initial seed selection, we conducted an ablation experiment comparing human-curated seeds with teacher-generated seeds created entirely from scratch. Specifically, we instructed the Teacher model to generate 100 seed problems without any human intervention, using only high-level domain specifications (e.g., "generate diverse mathematical problems covering algebra, geometry, and number theory").

Table 16 presents the Stage 1 results on MATH-500 benchmark. The negligible performance difference (0.1 percentage points) demonstrates that our framework's success is driven primarily by the co-evolutionary mechanism rather than the quality or source of initial seeds. This finding supports the data-free bootstrapping argument used in our framing: after the one-time cold-start SFT, the co-evolution loop can be initialized with fully synthetic seeds rather than a human-curated problem list.

Table 16: Comparison of Seed Problem Sources (Stage 1 Results on MATH-500)

| **Seed Source** | **MATH-500 Score** | **Difference** |
|---|---|---|
| Human-curated seeds (Original) | 60.2% | - |
| Teacher-generated seeds (From-Zero) | 60.1% | $\downarrow$0.1 |

We have open-sourced both the human-curated and teacher-generated seed lists, along with the complete generation protocol, to ensure full transparency and reproducibility of our results.

## LIMITATIONS

The method remains dependent on a Teacher for verification and problem refinement during bootstrap, and the Teacher's inference cost can be large. References for refined problems are retained by same-Teacher consistency filtering rather than an independent verifier, so errors can propagate through the curriculum. The current evaluation emphasizes domains with an objective correctness signal, and the observed saturation does not constitute a formal convergence guarantee.

## AI USE STATEMENT

The authors did not use generative AI tools to produce the manuscript, figures, or experimental results.

## REPRODUCIBILITY STATEMENT

Implementation details, prompts, hyperparameters, evaluation protocols, and additional ablations are provided in the appendix. The source archive contains the figures, bibliography, and all section files used to build this manuscript.