StudyBench: Can Self-Evolution Squeeze Textbooks for Olympiad Capability?

Anonymous ACL submission

Abstract

Humans need only studying a handful of well-written textbooks to master a discipline and attempt its hardest problems. We argue that an ideal self-evolution method should share the same property, that is **autonomously learning from limited training material for transferable problem-solving capability**. However, we still lack a direct measurement for it. We introduce **StudyBench**, a controlled physics benchmark that directly measures how efficiently a self-evolution method converts training material into capability. We organise test set into an **Application Set**, consisting of difficult textbook problems and evaluating absorption ability and a **Transfer Set**, consisting of olympiad-level problems and evaluating transfer ability. Benchmarking representative self-evolution methods spanning synthetic data, self-play, and context evolution against a fixed base model, we find that they absorb the training material similarly well but separate three-fold on the harder Transfer Set. This exposes a **Guidance Gap**: even the strongest method closes less than half of what the same material unlocks when supplied as in-context guidance. Every method additionally hits a **Compute Plateau**, saturating well before exhausting its compute budget. The remaining gap is therefore a method problem rather than a data or compute problem. By offering a clean and controlled benchmark, StudyBench turns self-evolution progress from an open-ended pursuit into a measurable target for future research.

1 Introduction

Self-evolution, the capacity for a model to keep improving on its own without being capped by limited supply of high-quality data, shows the promise of true Artificial General Intelligence (Goertzel and Pennachin, 2007; Silver and Sutton, 2025). We argue that any system worthy of that promise must succeed at two things at once. First, it should continuously *absorb* new knowledge from its environment (Yuan et al., 2026). Second, and more critically, it should continuously *evolve* absorbed knowledge into transferable problem-solving capability, instead of merely memorizing or paraphrasing what it has seen (Mirzadeh et al., 2025; Huang et al., 2025b). The second ability is arguably the harder of the two: absorption alone is bounded by what the training material already states; yet real-world problems seldom have direct precedents in the training material, and addressing them therefore demands *transferable* problem-solving capability. Whether self-evolution can scale beyond the data it consumes therefore turns on how efficiently it performs this **knowledge-to-capability conversion**.

Despite a rapidly growing literature (Gao et al., 2025; Novikov et al., 2025; Huang et al., 2025a) on self-evolution, we still lack a clear way to measure how effectively a method performs this conversion. Static high-difficulty exams such as AIME and Humanity's Last Exam (Phan et al., 2025) only give a final score, conflating the algorithm's contribution with that of the training data and the base model. Dynamic and lifelong-learning benchmarks (Castillo-Bolado et al., 2024; Zheng et al., 2025a; Wan and Ma, 2025; Dou et al., 2026) test whether a model reuses earlier experiences to do better on later problems; they address local adaptation, while we operate at a higher level, testing conversion from experiences to transferable capability. The deeper reason these existing evaluations fall short is that the conversion itself is hard to measure directly: any such measurement faces three obstacles. First, **the vanishing capability gap**: when the base model already passes the test set on its own, any post-training score merely surfaces existing capability rather than what the algorithm added. Second, **unreachable targets**: when the training material is insufficient for any algorithm to derive the test solutions. Third, **confounded attribution**: when the base model, the training material, and the

algorithm vary together, thus the score conflates all three. To our knowledge, no existing evaluation eliminates all three at once.

We close this gap with **StudyBench**, aiming to directly reveal the enhancement brought by self-evolution methods. We instantiate the setting in physics: 11 canonical physics textbooks serve as the training material, factored into three layers to fit requirements of different self-evolution methods: a **Corpus** of raw passages, **Instructions with Answer**, and **Instructions without Answer**. The training material is paired with two complementary test splits that form a built-in difficulty progression. The **Application Set** consists of difficult end-of-chapter exercises drawn from the textbooks themselves, retained only when the base model fails to solve them on its own; it probes whether an algorithm has absorbed the training material. The **Transfer Set** consists of olympiad-level theory problems, retained only when the base model alone fails but succeeds under a textbook-grounded guidance trace built from passages taken directly from the training material; it probes whether the method has additionally converted that material into capability transferable to problems harder than any in the textbooks themselves.

This construction guarantees three properties by design: **(1) Capability Gap**, that every retained test problem falls strictly outside the base model's capability, attributing scores on test set to improvement of the model's ability; **(2) Reachability**, that every retained test problem is solvable by utilizing the training material, eliminating unreachable targets; and **(3) Controlled Attribution**, that the base model, the training material, and the evaluation protocol are identical across all experiments, eliminating confounded attribution. The score is therefore a direct measure of each method's knowledge-to-capability conversion, allowing methods from different families to meet on a single, comparable leaderboard.

On StudyBench, we benchmark five representative self-evolution methods against a fixed base model and find that **(i)** all five absorb the training material similarly well—Application-Set sub-problem gains cluster between $+12.47$ and $+15.91$—so absorption is no longer the discriminator; **(ii)** on the harder Transfer Set the same methods spread three-fold ($+2.31$ to $+7.65$ Sub.), and even the strongest recovers only $\approx 40\%$ of the headroom that a textbook-grounded guidance trace already provides from the same material; and **(iii)** every method plateaus well before exhausting its compute budget. The remaining gap is therefore a method problem rather than a data or compute problem, leaving substantial room for future work.

2 StudyBench

StudyBench is inspired by the way humans master a discipline: a motivated student typically needs no more than a handful of well-chosen textbooks and some deliberate practice to attempt the hardest problems in the field. A self-evolution method, placed in an analogous setting, ought to demonstrate comparable capability. StudyBench builds such a setting and provides a means to measure it. We instantiate it in physics, where final answers are verifiable, the set of standard textbooks is small and widely shared across physics curricula of top universities and olympiad training, and hard problems require both textbook knowledge and the reasoning capability to use it. These properties make physics an ideal setting for measuring capability rather than knowledge alone.

2.1 Benchmark Construction

Figure 1 summarises the construction. Each textbook yields three nested layers of training material: the **Corpus** of raw passages contains the **Instructions without Answer** (every exercise as a problem statement), which in turn contains the **Instructions with Answer** (the subset whose answers the textbook supplies). Each self-evolution method draws on whichever layer fits its training paradigm. The test set, in turn, draws from two complementary sources of escalating difficulty: difficult end-of-chapter problems from these same textbooks (Table 5) and problems from six international physics and astronomy olympiads (Table 6). From both pools we keep only problems on which the base model fails; for the olympiad pool we additionally retain only those the base model can solve under a textbook-grounded guidance trace, certifying that the answer is reachable from the training material. The retained textbook problems form the **Application Set**, which measures whether a method has absorbed the training material well enough to apply it where it was first introduced; the retained olympiad problems form the **Transfer Set**, which measures whether the method has additionally converted that material into capability that transfers to problems harder than any in the textbooks themselves. The remainder of this subsection details

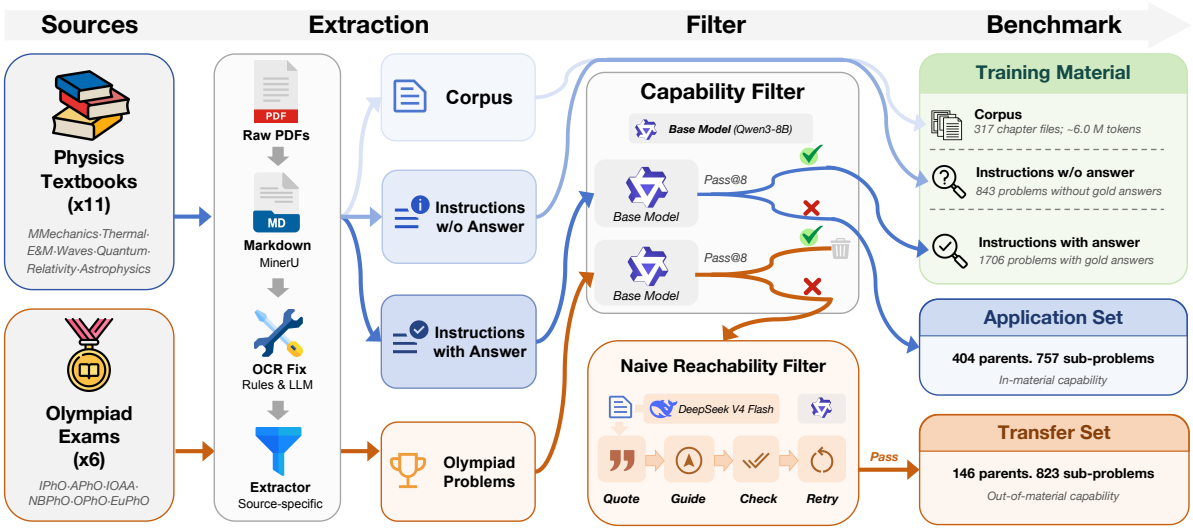

Figure 1: Construction pipeline for StudyBench. Textbooks are extracted into a *Corpus*, *Instructions with Answer* , and *Instructions without Answer*, which together form the training material; olympiad problems are extracted along a separate, test-only stream. A shared *Capability Filter* keeps only items the base model fails on. Failed textbook problems become the **Application Set**; failed olympiad problems pass an additional *Naive Reachability Filter* and become the **Transfer Set**.

each step.

Sources. The 11 textbooks (Table 5) are each currently adopted as a course text at top universities and independently recommended either by olympiad-training coaches or on the competitions' own preparation pages. By sub-discipline, the 11 textbooks jointly cover the syllabus of all six olympiads.

Extraction. All source materials are PDFs. We convert every PDF to Markdown via MinerU (Wang et al., 2026) and then fix common OCR errors (broken super- and subscripts, malformed LaTeX, dropped figure captions) with by first applying deterministic rules and then using an LLM to handle cases the rules cannot. With Claude Opus 4.7 as a coding assistant, we then write a separate extractor for each textbook and each competition, since each source has its own layout. From each textbook we extract its worked examples and end-of-chapter exercises, and from each competition we extract every theory problem of every past edition. Physics problems are commonly split into several sub-problems sharing a common setup, so each extracted record stores both the full problem statement and a list of per-sub-problem entries carrying the sub-problem text, the reference solution where given, and the gold answer. The gold answer of every sub-problem is further classified by DeepSeek V4 Flash (DeepSeek-AI, 2026) into one of nine answer types: `NV` (numeric value), `EX` (symbolic expression), `EQ` (equation), `TUP` (ordered tuple), `IN` (interval), `MC` (single multiple-choice letter), `TF` (boolean), `QL` (short qualitative phrase), and `ALT` (alternative acceptable forms of one answer); composite types (`TUP` and `ALT`) additionally carry a per-position sequence that tells the verifier how to judge each slot. Appendix C gives the full record schema and shows one example.

Capability Filter. We run the fixed base model with pass@8 sampling on both the textbook and competition pools. Solved textbook problems are retained in the training material, while failed textbook problems directly form the Application Set. Since the prerequisite material lives in the same chapter, Application Set is reachable by construction. Failed competition problems advance to the Naive Reachability Filter; solved competition problems are discarded.

Naive Reachability Filter. Since we cannot precisely characterise the reachability boundary, we adopt a naive but auditable proxy: a competition problem is admitted only if the base model can solve it under a teacher-distilled, textbook-grounded guidance. The procedure is driven by a strong teacher **DeepSeek V4 Flash** (DeepSeek-AI, 2026) and consists of four steps:

- **Quote.** For each relevant textbook chunk, the teacher extracts verbatim passages supporting the gold solution, such as formulae, lemmas, analogous worked examples, refusing to fabri-

cate any sentence absent from the chunk.

- **Guide.** Given the deduplicated quotes and the problem statement (but *not* the gold solution), the teacher composes a step-by-step guidance grounded only in those quotes.
- **Ground-check.** A separate prompt re-reads quotes and guidance and flags any claim unsupported by the quotes; failed guidances are regenerated up to twice.
- **Retry.** We admit the problem into the Transfer Set if the base model solves it at least once in eight attempts under this grounded guidance.

Together these four steps form an explicit, textbook-auditable witness that the gold answer is reachable by recombining the training material.

Properties. By construction, the resulting benchmark satisfies three properties. **(1) Capability Gap**, that every retained test problem falls strictly outside what the base model can already do, attributing scores on the test set to capability genuinely gained from the training material rather than residual base ability; **(2) Reachability**, that every retained test problem is solvable by recombining content from the training material (by construction in the Application Set, and empirically certified by the textbook-grounded guidance in the Transfer Set), eliminating unreachable targets; and **(3) Controlled Attribution**, that the base model, the training material, and the evaluation protocol are fixed across submissions, so a method's score reflects only what the method does between them.

2.2 Evaluation

Protocol. Each method is free to draw on any subset of the training material (Corpus, Instructions with Answer, or Instructions without Answer, in any combination) to suit its training paradigm. We then evaluate every evolved model on both test sets under a unified decoding protocol: $k = 8$ samples per sub-problem at temperature 1.0, top-p 0.95, top-k 20, with a 32,768-token cap. From these samples we report two accuracies. Let $\mathcal{P}$ denote the parent problems in a test set, n_p the number of sub-problems of parent p, and $c_{p,j,a} \in \{0, 1\}$ the verifier's judgement on the j-th sub-problem of parent p in attempt a. **Parent accuracy** ($\mathrm{Par@}k$) counts a parent correct only if some single attempt solves every one of its sub-problems:

$$\mathrm{Par@}k = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \max_{1 \le a \le k} \prod_{j=1}^{n_p} c_{p,j,a}.$$

Sub-problem accuracy ($\mathrm{Sub@}k$) flattens parents into sub-problems and marks each correct if any of the k attempts solves it:

$$\mathrm{Sub@}k = \frac{1}{\sum_{p \in \mathcal{P}} n_p} \sum_{p \in \mathcal{P}} \sum_{j=1}^{n_p} \max_{1 \le a \le k} c_{p,j,a}.$$

Sub-problem Evaluation. Multi-part problems are scored sub-problem by sub-problem with conversational continuity: at sub-problem i, the model sees the shared stem, the $i - 1$ prior sub-problem statements, and the *model's own* prior answers to them, but never the gold solutions. When the model fails to produce a final answer for sub-problem i, we insert a fixed placeholder stating the failure in the assistant slot and continue with sub-problem $i + 1$ rather than discarding the rest of the parent. This isolates each sub-problem's correctness, so $\mathrm{pass@}k$ can be reported per sub-problem and aggregated per parent.

Verifier. We build our verifier on the rule-based judger of UG-Physics (Xu et al., 2025), adding one new primitive answer type `QL` (short qualitative phrases such as "tidal forces"). We also introduce a new field, `type_sequence`, which lets the verifier dispatch composite answers (`TUP` and `ALT`) onto position-wise primitive judgers. The resulting verifier applies a different judging rule for every one of the nine answer types. When the verifier is wired into an RL-based self-evolution method as a reward signal, we expose only this rule-based part to avoid reward hacking. For leaderboard evaluation we additionally route failed problems to **DeepSeek V4 Flash** as an LLM judger, typically because the prediction contains unparseable LaTeX; the full judge prompt is reproduced in Appendix D.

2.3 Contamination

The 11 textbooks and the Olympiad archives are public, so a problem's statement or answer can, in principle, leak into a method's pipeline at two stages: (a) the *pretraining* of the base model, or (b) *training material* of a self-evolution method. StudyBench neutralises both by design.

(i) Pretraining leakage is screened out by the Capability Filter. A problem enters either test set only if the base model fails it on all eight attempts under our decoding protocol. Any answer the base model has memorised well enough to recover is therefore removed from the test sets by construction, regardless of whether the problem statement appears verbatim in pretraining.

(ii) Training-material leakage is removed by redaction. Transfer Set problems come from olympiad theory exams, which do not appear in any of the eleven textbooks that constitute the training material. Application Set problems do come from those textbooks, but for every retained parent we excise both the problem statement and the reference solution (including any back-of-book answer key or solution-manual entry) from the raw markdown before assembling the training material, and we audit the resulting corpus to confirm no verbatim residue remains; Appendix E details the two-pass redaction pipeline and the audit procedure. A method that memorises every page of its training material therefore gains access to none of the answers it will be asked for.

2.4 Statistics

Test sets. StudyBench provides a total of 1580 sub-problems. The filters retain 404 Application Set parents (757 sub-problems) and 146 Transfer Set parents (823 sub-problems); Table 1 gives the per-source breakdown. Together, the two sets let us measure both **in-material** capability and **out-of-material** capability. Three of the eleven textbooks (*The Physics of Waves*, *Concepts in Thermal Physics*, and *Introduction to Electrodynamics*) contribute no Application Set parents; we provide detailed reasons in Appendix B.

Training material. Table 2 reports the per-layer counts of the Corpus, the Instructions without Answer, and the Instructions with Answer. The three layers cover the supervision regimes of the major self-evolution families.

Table 1: Per-source breakdown of the two test sets retained after filtering.

Source	Parents	Sub-probs
Application Set		
Modern Astrophysics	174	380
Classical Mechanics	61	83
Quantum Physics	56	107
Schaum's Astronomy	43	52
Electricity and Magnetism	36	56
Special Relativity	24	44
Spacetime Physics	7	29
Mechanics	3	6
Total	**404**	**757**
Transfer Set		
IPhO	55	390
APhO	35	179
IOAA	22	116
NBPhO	22	70
OPhO	11	66
EuPhO	1	2
Total	**146**	**823**
Combined	**550**	**1,580**

Table 2: Statistics of the three nested training-material layers. *Instructions with Answer* means we successfully extracted the instruction-answer pair, while *Instructions without Answer* means we failed to find the corresponding answer

Layer	Count
Corpus chapter files	317
total size (MB)	18.19
total tokens	$\sim$ 6.0M
Instructions without Answer	843
Instructions with Answer	1,706

3 Experiments

Baselines. We benchmark a representative set of self-evolution methods, grouped by which layer of the training material each one consumes. **(1) Corpus. Bonito** (Nayak et al., 2024) runs a task-conditioned generator over textbook passages to synthesise question–answer pairs and performs supervised fine-tuning on the base model with them. **(2) Instructions with Answer. GEPA** (Agrawal et al., 2025) treats them as a development set and evolves a system prompt via reflective genetic search, while **ACE** (Zhang et al., 2025) distils them into an in-context playbook of formulae, strategies, and common pitfalls; both artefacts are injected into the system message at inference time without any weight update. **(3) Instructions without Answer. Intuitor** (Zhao et al., 2025) runs GRPO on the base model with these problem, deriving its reward from the policy's own self-certainty instead of gold answer. **(4) Data-free. R-Zero** (Huang et al., 2025a) bootstraps capability from Challenger-Solver self-play under self-consistency rewards. We include it to see how well a method that uses no training data at all can do on StudyBench.

Setup. We use **Qwen3-8B** with thinking mode enabled as the base model across all baselines. For each baseline we reproduce from its official repository and make only minimal modifications to fit StudyBench's training material and evaluation protocol. All self-evolution methods runs on a single 8×NVIDIA-A100-80GB node. Appendix G lists per-method modifications and full configurations.

Results. Table 3 reports the main results. Both parent and sub-problem accuracy are shown; because the base model has Par. $= 0.00$ on both test sets by construction, the Δ column applies only

Table 3: Main results on StudyBench. **Data** indicates which training-material layer each method consumes: *Corpus*, *Instructions without Answer* (Instr w/o Ans), *Instructions with Answer* (Instr w/ Ans), or *none* (—). **Par.** is parent accuracy (at least one of 8 attempts solves every sub-problem of the parent within that single attempt); **Sub.** is sub-problem accuracy (any of 8 attempts solves the sub-problem). Both are pass@8 under the unified decoding protocol of Section 2.2. We report a Δ column only for Sub., since M_0's Par. is 0.00 on both test sets by construction and ΔPar. would coincide with Par.; the trained methods' absolute Par. is shown alongside. All numbers in %.

		Application Set			Transfer Set		
	Data	Par.	Sub.	ΔSub.	Par.	Sub.	ΔSub.
Qwen3-8B	—	0.00	28.61	–	0.00	44.23	–
Self-evolution methods							
Bonito (Nayak et al., 2024)	Corpus	24.01	44.52	+ 15.91	5.48	51.88	+7.65
GEPA (Agrawal et al., 2025)	Instr w/ Ans	23.02	41.88	+ 13.27	2.74	47.63	+3.40
ACE (Zhang et al., 2025)	Instr w/ Ans	14.36	41.08	+12.47	4.11	46.54	+2.31
Intuitor (Zhao et al., 2025)	Instr w/o Ans	25.00	41.61	+ 13.00	3.42	48.85	+4.62
R-Zero (Huang et al., 2025a)	None	15.59	42.54	+13.93	3.42	48.00	+3.77

to Sub. On the Application Set, all five baselines achieve double-digit sub-problem gains over M_0's 28.61% (+12.47 to +15.91), and every method lifts parent accuracy into the 14–25% range, so each one absorbs the training material well enough to solve some previously unreachable textbook problems end-to-end. The picture on the Transfer Set is more differentiated: every method improves in sub-problem accuracy (+2.31 to +7.65), but **Bonito** leads on both metrics (+7.65 Sub.; 5.48 Par.), while **ACE** trails (+2.31 Sub.). Section 4.1 introduces a guidance ablation that explains this spread through the **Guidance Gap**—the accuracy a method recovers when textbook-grounded guidance is supplied at inference time but not internalised by training.

4 Analysis

4.1 The Guidance Gap (RQ1)

The Application Set probes whether a method has *absorbed* the training material; the Transfer Set probes whether that material has *transferred* to problems strictly harder than any in the textbooks. The two probes diverge sharply in Table 3: on the Application Set the five baselines gain a similar +12.47 to +15.91 Sub., so absorption is no longer the discriminator, but on the Transfer Set the same five spread three-fold (+2.31 to +7.65 Sub.) despite their comparable Application gains. **RQ1** therefore asks how much of the corpus-reachable knowledge certified by our Naive Reachability Filter has actually migrated into standalone transfer capability, rather than remaining merely reachable.

Guidance ablation. To answer this, we re-run the Transfer Set under the textbook-grounded guidance traces produced during construction (§2.1), applying them at inference. Our compute budget limits the ablation to M_0, GEPA, Intuitor, and Bonito (Table 4); ACE and R-Zero are discussed below using only their solo Transfer numbers from Table 3, and we are explicit about which claims do and do not extend to them. We define the **Guidance Gap** as the accuracy a method recovers when grounded guidance is supplied at inference but not internalised by training:

Table 4: Transfer-Set accuracy with textbook-grounded guidance, for the four baselines we re-ran under guidance. Solo accuracies repeat Table 3; guided accuracies use the same traces as the Naive Reachability Filter, injected at inference. **Guidance Gap** = $\text{Acc}_{\text{guided}} - \text{Acc}_{\text{solo}}$, reported for both metrics. M_0's guided Par. exceeds 0 because the Capability Filter (§2.1) rejects parents only on *solo* pass@8.

	Par.			Sub.		
Method	Solo	Guided	Gap	Solo	Guided	Gap
Qwen3-8B	0.00	10.27	10.27	44.23	61.60	17.37
GEPA	2.74	10.96	8.22	47.63	58.20	10.57
Intuitor	3.42	10.96	7.54	48.85	60.39	11.54
Bonito	5.48	12.33	6.85	51.88	62.33	10.45

$$\text{Guidance Gap} = \text{Acc}_{\text{guided}} - \text{Acc}_{\text{solo}}, \quad (1)$$

reported for both Par. and Sub. on the Transfer Set. Closing the gap means the method has converted into standalone capability what, in M_0, only an in-context guidance trace could surface.

A near-stable guidance ceiling. Once guidance is supplied, Sub. accuracy falls into a narrow 58.20–62.33% band and Par. accuracy into 10.27–12.33%. Training only mildly perturbs this ceiling—Bonito sits 0.73 Sub. points above M_0 while GEPA and

Intuitor sit slightly below—and even at the ceiling performance is far from saturation, since the Reachability Filter only certifies that each parent is solved at least once under guidance. The Guidance Gap therefore functions as an externally validated upper bound against which solo Transfer performance can be referenced.

Solo gains as a fraction of the ceiling. Read against this upper bound, each method's solo Transfer gain is the share of M_0's original Guidance Gap it has closed. On Sub., M_0's 17.37-point gap drops to 10.45 under Bonito (closure **40**%), 10.57 under GEPA (**39**%), and 11.54 under Intuitor (**34**%); on Par., M_0's 10.27-point gap drops to 6.85 under Bonito (**33**%), 7.54 under Intuitor (**27**%), and 8.22 under GEPA (**20**%). Both metrics rank Bonito first, while the Application Set—where the same baselines cluster within 3.44 Sub. points—cannot distinguish any of this; the two test levels are complementary by design.

Substantial headroom remains. Bonito's $\approx 40\%$ Sub. and $\approx 33\%$ Par. closure leave most of M_0's Guidance Gap unrecovered. Because training barely moves the ceiling, what remains is an *internalisation* problem rather than a *reachability* problem: the same training material is already sufficient to surface every retained parent under guidance, yet no learned method recovers more than half of what guidance unlocks.

4.2 The Compute Plateau (RQ2)

RQ1 left most of the Guidance Gap unclosed: even Bonito, the strongest method, recovered at most $\approx 40\%$ of M_0's Sub. gap on the Transfer Set. **RQ2** asks whether the remaining gap can be closed by simply running each self-evolution loop longer.

Setup. For each baseline, we evaluate its model checkpoint (Bonito, Intuitor, R-Zero) or inference-time artefact (GEPA, ACE) at a sequence of intermediate snapshots during the self-evolution loop, and plot Application-Set sub-problem accuracy as a function of cumulative GPU-time on a single $8\times$ A100-80GB node. Total compute spans nearly two orders of magnitude across methods — ~ 6.5 GPU-hr for Bonito to ~ 265 GPU-hr for R-Zero — so the five curves cannot be meaningfully overlaid on a shared axis. We therefore show Bonito in the main text (Figure 2) and reproduce the four other curves individually in Appendix F.

The plateau is universal. Every method saturates well before exhausting its compute budget and then

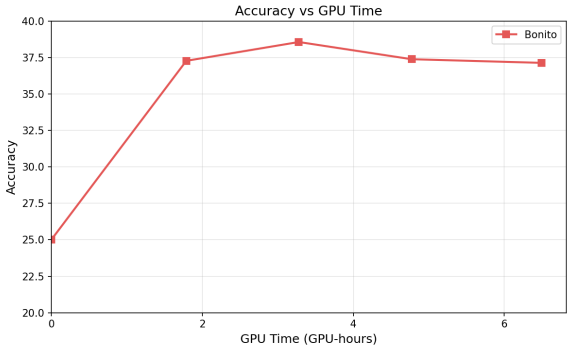

Figure 2: Application-Set sub-problem accuracy of **Bonito** as a function of cumulative self-evolution GPU-time on a single $8\times$ A100-80GB node. Bonito reaches 37.2% within 1.8 GPU-hr (the first $\sim 28\%$ of its budget) and the remaining 4.7 GPU-hr never lift the curve above the 3.2-hr peak; the same plateau pattern holds for the other four baselines (Appendix F).

stays flat or drifts downward. Bonito hits 37.2% at 1.8 GPU-hr ($\sim 28\%$ of its budget); ACE hits 36.9% at 3 GPU-hr ($\sim 13\%$); Intuitor hits 37.5% at 16 GPU-hr ($\sim 29\%$); GEPA hits 34.5% at 12 GPU-hr ($\sim 13\%$); R-Zero hits 36.9% at ~ 150 GPU-hr ($\sim 57\%$). After this saturation point the trajectories never resume climbing: the largest peak-to-end drops are Intuitor (-3.3 Sub., from 37.8% at 46 hr to 34.5% at 56 hr), R-Zero (-1.5, from 37.3% to 35.8% over the final ~ 75 hr), and Bonito (-1.4, from 38.5% at 3.2 hr to 37.1% at 6.5 hr); GEPA oscillates within a 4.5-point band (31.2%–35.7%); and ACE moves only $+0.4$ points over its final ~ 21 GPU-hr. None of the late phases looks like continued learning — the most charitable reading of any of them is "noisy plateau".

Compute is not the bottleneck. The remaining headroom from RQ1 therefore cannot be closed by running any of these loops longer. At each method's converged snapshot, the Guidance Gap of Table 4 is already in place, and extending the loop by $1.8\times$ (R-Zero) to nearly $8\times$ (ACE, GEPA) past saturation moves Application accuracy by less than a point in either direction. The Guidance Gap is not a budget problem but a method problem: closing more of it will require a different self-evolution loop, not a longer one.

Compute spread does not buy accuracy spread. Across the five baselines, total compute spans $\sim 40\times$ (6.5 to 265 GPU-hr) yet peak Application accuracy spans only ~ 3 points (35.7% to 38.5%). R-Zero is the extreme of this trade-off: it spends roughly $40\times$ Bonito's wall-clock budget to land

within 1.2 points of Bonito's peak. Reading R-Zero's data-free property as "free because it consumes no data" would therefore be misleading; the absence of a data dependency is paid for, in full, on the compute side, while the resulting capability sits on the same plateau every other method reaches.

5 Related Work

5.1 Self-Evolving Methods

Self-play and intrinsic rewards. R-Zero (Huang et al., 2025a) (our data-free baseline) and Absolute Zero (Zhao et al., 2026) alternate a Challenger and a Solver under self-consistency rewards; SPICE (Liu et al., 2025) grounds the Challenger in a document corpus to combat hallucination amplification. Self-Rewarding LMs (Yuan et al., 2024) run iterative DPO with the policy as its own judge, and Intuitor (Zhao et al., 2025) (our Instructions-without-Answer baseline) replaces the judge with the policy's self-certainty in GRPO.

Context evolution and self-editing. GEPA (Agrawal et al., 2025) and ACE (Zhang et al., 2025), our two Instructions-with-Answer baselines, respectively evolve a system prompt by reflective genetic search and distil instructions into an in-context playbook. SEAL (Zweiger et al., 2026) trains the model to emit "self-edits" that turn each passage into finetuning data and updates the edit policy by RL.

Memory and test-time evolution. Voyager (Wang et al., 2023), Reflexion (Shinn et al., 2023), ExpEL (Zhao et al., 2024), and ReasoningBank (Ouyang et al., 2025) accumulate per-trajectory memory at test time, while AlphaEvolve (Novikov et al., 2025) applies evolutionary search at inference.

Across these families, methods are reported on open-ended benchmarks with implicit or freely chosen training data, making it impossible to attribute gains to a specific corpus or compare methods from different families on equal footing.

5.2 Benchmarks for Self-Evolution

Dynamic and lifelong-learning benchmarks. Several benchmarks measure how a fixed model accumulates knowledge or skills over a stream of interactions. The LTM Benchmark (Castillo-Bolado et al., 2024) runs a long interleaved user–agent dialogue; LifelongAgentBench (Zheng et al., 2025a) asks agents to accumulate atomic skills across task sequences; StoryBench (Wan and Ma, 2025) casts evaluation as branching interactive fiction; EvaLearn (Dou et al., 2026) organises problems into task-typed sequences and quantifies within-sequence transfer. The unit of comparison is local adaptation within the stream, not how much of an external corpus a method has internalised, and training data is typically chosen by the submitter rather than fixed.

Benchmarks specifically for self-evolution. SE-Bench (Yuan et al., 2026) measures knowledge internalisation; NewtonBench (Zheng et al., 2025b) measures generalisable scientific-law discovery in LLM agents; and Frontier-Eng (Chi et al., 2026) measures self-evolving agents on real-world engineering tasks with generative optimisation. These come closest to StudyBench's question, but none jointly enforces the three desiderata of §2—capability gap, reachability, and controlled attribution—nor provides a built-in difficulty progression with an externally validated guidance ceiling against which solo capability can be referenced. StudyBench fills exactly that gap.

6 Conclusion

We introduce **StudyBench**, a controlled physics benchmark that directly measures how efficiently a self-evolution method converts a fixed corpus into transferable problem-solving capability. By pairing 11 canonical textbooks with two complementary test levels—an Application Set of failed end-of-chapter exercises and a Transfer Set of olympiad problems certified reachable under textbook-grounded guidance—StudyBench simultaneously eliminates the three obstacles of vanishing capability gap, unreachable targets, and confounded attribution. On the resulting leaderboard, five representative self-evolution methods absorb the training material indistinguishably well yet separate three-fold on transfer: the strongest recovers only $\approx 40\%$ of the Sub. headroom that a guidance trace already provides from the same material, and every method plateaus well before exhausting its compute budget. The remaining gap is therefore a method problem rather than a data or compute problem—a distinction StudyBench is designed to make directly measurable for future self-evolution research.

Limitations

All experiments fix the base model to Qwen3-8B with thinking mode and instantiate the setting in physics over 11 textbooks; while the construction principles—Capability Filter, Naive Reachability Filter, and two-level test design—are domain-agnostic, we have not verified that our specific findings on the Guidance Gap and the compute plateau replicate under other base models or disciplines. Compute constraints additionally limit the guidance ablation (Table 4) to four of the five baselines; ACE and R-Zero are positioned only by their solo Transfer numbers, so our closure-rate claims on the Guidance Gap directly extend to four baselines rather than all five.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, and 1 others. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

David Castillo-Bolado, Joseph Davidson, Finlay Gray, and Marek Rosa. 2024. Beyond prompts: Dynamic conversational benchmarking of large language models. *Advances in Neural Information Processing Systems*, 37:42528–42565.

Yizhe Chi, Deyao Hong, Dapeng Jiang, Tianwei Luo, Kaisen Yang, Boshi Zhang, Zhe Cao, Xiaoyan Fan, Bingxiang He, Han Hao, and 1 others. 2026. Frontier-eng: Benchmarking self-evolving agents on real-world engineering tasks with generative optimization. *arXiv preprint arXiv:2604.12290*.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Shihan Dou, Ming Zhang, Chenhao Huang, Jiayi Chen, Feng Chen, Shichun Liu, Yan Liu, Chenxiao Liu, Cheng Zhong, Zongzhang Zhang, and 1 others. 2026. Evalearn: quantifying the learning capability and efficiency of llms via sequential problem solving. *Advances in Neural Information Processing Systems*, 38:125892–125945.

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

Ben Goertzel and Cassio Pennachin. 2007. *Artificial general intelligence*, volume 2. Springer.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025a. R-zero: Self-evolving reasoning LLM from zero data. *Preprint*, arXiv:2508.05004.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, and 1 others. 2025b. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. *arXiv preprint arXiv:2502.06453*.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. 2025. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pages 94743–94765.

Nihal Nayak, Yiyang Nan, Avi Trost, and Stephen Bach. 2024. Learning to generate instruction tuning datasets for zero-shot task adaptation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12585–12611.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, and 1 others. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

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

David Silver and Richard S Sutton. 2025. Welcome to the era of experience. *Google AI*, 1:11.

Luanbo Wan and Weizhi Ma. 2025. Storybench: A dynamic benchmark for evaluating long-term memory with multi turns. *arXiv preprint arXiv:2506.13356*.

Bin Wang, Tianyao He, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Tao Chu, Yuan Qu, Zhenjiang Jin, Weijun Zeng, Ziyang Miao, and 1 others. 2026. Mineru2. 5-pro: Pushing the limits of data-centric document parsing at scale. *arXiv preprint arXiv:2604.04771*.

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

Xin Xu, Qiyun Xu, Tong Xiao, Tianhao Chen, Yuchen Yan, Jiaxin Zhang, Shizhe Diao, Can Yang, and Yang Wang. 2025. Ugphysics: A comprehensive benchmark for undergraduate physics reasoning with large language models. *arXiv preprint arXiv:2502.00334*.

Jiarui Yuan, Tailin Jin, Weize Chen, and Zeyuan Liu. 2026. Se-bench: Benchmarking self-evolution with knowledge internalization. *Preprint*, arXiv:2602.04811.

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

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, and 1 others. 2025. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 17, pages 19632–19642.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2026. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, (17):105816–105879.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. 2025. Learning to reason without external rewards. *Preprint*, arXiv:2505.19590.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. 2025a. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*.

Tianshi Zheng, Kelvin Kiu-Wai Tam, Newt Hue-Nam K Nguyen, Baixuan Xu, Zhaowei Wang, Jiayang Cheng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, and 1 others. 2025b. Newtonbench: Benchmarking generalizable scientific law discovery in llm agents. *arXiv preprint arXiv:2510.07172*.

Adam Zweiger, Jyo Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. 2026. Self-adapting language models. *Advances in Neural Information Processing Systems*, 38:74084–74115.

A Source materials

Table 5 lists the eleven physics textbooks that make up the training material $\mathcal{C}$, alongside the sub-discipline each one covers. Table 6 lists the six international physics and astronomy olympiads that contribute to the competition pool, together with the year range we collected. Together the two tables make explicit the syllabus-level match between $\mathcal{C}$ and the test pool: every sub-discipline labelled in Table 5 is also a sub-discipline tested by at least one olympiad in Table 6.

Table 5: The 11 physics textbooks comprising StudyBench's training material. Where an official solution manual exists, it is bundled alongside the textbook.

Textbook	Sub-discipline
Introduction to Classical Mechanics	Mechanics
Introduction to Mechanics	Mechanics
Electricity and Magnetism	Electromagnetism
Introduction to Electrodynamics	Electromagnetism
Concepts in Thermal Physics	Thermal Physics
The Physics of Waves	Waves
Quantum Physics	Quantum Physics
Special Relativity	Relativity
Spacetime Physics	Relativity
An Introduction to Modern Astrophysics	Astrophysics
Schaum's Outline of Astronomy	Astrophysics

Table 6: The six international physics olympiads we utilized.

Acronym	Full name
APhO	Asian Physics Olympiad
EuPhO	European Physics Olympiad
IPhO	International Physics Olympiad
IOAA	Intl. Olympiad on Astronomy & Astrophysics
NBPhO	Nordic-Baltic Physics Olympiad
OPhO	Online Physics Olympiad (Invitational)

B Textbooks that contribute no Application Set parents

Three of the eleven textbooks in $\mathcal{C}$ contribute zero parents to the Application Set: Georgi's *The Physics of Waves*, Blundell & Blundell's *Concepts in Thermal Physics*, and Griffiths's *Introduction to Electrodynamics*. Their reasons are distinct, and in only one case does the explanation reduce to "M_0 already solves these problems".

The Physics of Waves (Georgi). This is the most compact textbook in $\mathcal{C}$ by a wide margin: across the entire book it contains only **43** end-of-chapter exercises, against 200–500 for every other textbook (e.g. **487** for Griffiths, **489** for Eisberg & Resnick, **352** for Morin). Even before any filtering, the candidate pool from this book is too small to survive a pass@8 Capability Filter with appreciable margin.

Concepts in Thermal Physics (Blundell & Blundell). The textbook itself yields a competitive **302** end-of-chapter exercises, but its accompanying solution manual is unusually condensed: a single 0.14 MB markdown file lists **204** solutions covering only a fraction of the textbook's exercises, with most chapters carrying 5–10 solutions out of $\sim$ 8–13 problems. Reliable answer-key extraction therefore yielded a much smaller pool of problems with verifiable gold answers than for the other contributors, and the curated input that reached the Capability Filter contained no Blundell parents.

Introduction to Electrodynamics (Griffiths). Griffiths's textbook contains **487** end-of-chapter exercises and ships a comprehensive 13-file chapter-by-chapter solution manual, so our pipeline successfully attached gold answers to **391** of them. Of these, only **9** ended up in the input set that the Capability Filter ran on, and M_0 passed all 9 at pass@8. Expanding the curated input to the remaining $\sim$ 380 verifiably-answered Griffiths problems is a clear target for a follow-up version of the benchmark.

C An extracted record

Each extracted record represents one physics problem in a unified schema. The top-level fields capture provenance (`source`, `source_problem_id`, `title`, `problem_type`) and the full `problem` statement. For multi-part problems, the record additionally holds a `sub_problems` list with one entry per sub-question; each entry carries:

- `problem_id` — a label such as `(a)` or `T1(b)` that disambiguates the sub-question within its parent;
- `problem` — the sub-question text;
- `solution` — the textbook's reference solution where one is provided;
- `answer` — the gold answer, with each final quantity wrapped in `\boxed{}`;
- `answer_type` — one of the nine answer types listed in Section 2.1, drawn from {`NV`, `EX`, `EQ`, `TUP`, `IN`, `MC`, `TF`, `QL`, `ALT`};
- `type_sequence` — for composite types (`TUP` and `ALT`), the inner answer type at each position; empty for the seven primitive types.

Solo problems lift these per-sub-problem fields to the top level and leave `sub_problems` empty.

Below is one extracted record — exercise P1.3 ("Force from a cone") from Purcell's *Electricity and Magnetism* — whose two sub-problems carry different answer types: sub-problem (a) yields a single equation (`EQ`), and sub-problem (b) yields an ordered tuple of two equations (`TUP` with `type_sequence = "EQ,EQ"`).

```
{
  "source": "Purcell_EM",
  "source_problem_id": "P1.3",
  "title": "Force from a cone",
  "problem_type": "Electromagnetic Fields",
  "problem": "(a) A charge $q$ is at the tip of
    a hollow cone with surface charge density $
    \\sigma$, slant height $L$, and half-angle $
    \\theta$. ... (b) If the top half of the
    cone is removed, what is the force on $q$,
    and at what $\\theta$ is it maximum?",
  "sub_problems": [
    {
      "problem_id": "(a)",
      "problem": "A charge $q$ is at the tip of
        a hollow cone with surface charge
        density $\\sigma$, slant height $L$, and
        half-angle $\\theta$. What can you say
        about the force on $q$ due to the cone?",

      "solution": "Integrating the per-ring
        vertical force over $x \\in (0, L]$
        gives $F \\propto \\int_{0}^{L} dx/x$,
        which diverges as $x \\to 0$.",
      "answer": "\\boxed{F = \\infty}",
      "answer_type": "EQ",
      "type_sequence": ""
    },
    {
      "problem_id": "(b)",
      "problem": "If the top half of the cone is
        removed, what is the force, and at what
        $\\theta$ is it maximum?",
      "solution": "The integral now runs from $L
        /2$ to $L$: $F = \\frac{q\\sigma\\ln 2}
        {4\\epsilon_0}\\sin 2\\theta$, which is
        maximised at $2\\theta = 90^\\circ$.",
      "answer": "\\boxed{F = \\frac{q\\sigma\\ln
        2} {4\\epsilon_0} \\sin 2\\theta},
        \\boxed{\\theta_{\\max} = 45^\\circ}",
      "answer_type": "TUP",
      "type_sequence": "EQ,EQ"
    }
  ]
}
```

D LLM-judge prompt

For the verifier's LLM-judge fallback (Section 2.2), we ask DeepSeek V4 Flash to compare a student-extracted answer against the reference answer and return a binary verdict together with a short justification. The full prompt template is reproduced below. The placeholders `{{problem}}`, `{{RS}}`, `{{RA}}`, `{{SS}}`, `{{SA}}` are filled in at run-time with the problem statement, the reference solution, the reference final answer(s), the student's full solution, and the answer(s) extracted from the student's solution, respectively.

```
You are an expert physics / astronomy grader.
You will be given

1. The original problem statement.
2. The reference (official) step-by-step
solution.
3. The reference final answer(s).
4. A student's full solution.
5. The answer(s) extracted from the student's
solution.

Your job is ONLY to decide whether the student's
 final answer is mathematically and physically
equivalent to the reference final answer, taking
 into account acceptable tolerances (e.g. unit
conversion, reasonable rounding, equivalent
algebraic forms, equivalent sign conventions,
equivalent multiple-choice letters, etc.).

You are not grading the derivation; if the final
 answer matches up to acceptable numerical
tolerance or algebraic equivalence, it is
correct, even if the reasoning was imperfect.
Conversely, a correct derivation that ends with
an incorrect numerical value should be judged
incorrect.

You MUST output your response in exactly the
following format, with nothing else before or
after:

## Equivalence Judgement
TRUE

## Justification
<one or two sentences explaining the decision>

Where the line after "## Equivalence Judgement"
is either the single word TRUE or the single
word FALSE (uppercase, no punctuation).

Guidelines for numerical answers:
- Accept answers within ~2% relative tolerance
for "order of magnitude" / "estimate" style
problems.
- Accept answers within the explicitly stated
tolerance range given in the reference solution,
 if any.
- Different but equivalent units (e.g. "20 kT"
vs "2e4 T", "90 m" vs "90 meters") are
equivalent.
- Different precisions that round to the same
value at the reference's stated precision are
equivalent.

Guidelines for symbolic answers:
- Expressions that are algebraically equal after
 simplification are equivalent.
- Expressions differing only by a named physical
 constant that has been replaced with its symbol
 (e.g. "c" vs "3e8 m/s") are equivalent.
- Expressions differing only by dimensionally-
trivial rearrangement (e.g. d * sqrt(d/GM) vs
sqrt(d^3/GM)) are equivalent.
```

```
Guidelines for tuples / multi-answer:
- Match element-wise; order matters unless the
problem clearly says otherwise.
- For ALT answer types (alternative acceptable
answers), matching ANY one of the listed
acceptable answers is sufficient.

Guidelines for True/False / multiple choice:
- Normalize case and synonyms ("Yes" == "True"
== "T", "No" == "False" == "F").

Guidelines for qualitative answers (short
natural-language phrases such as "tidal forces",
 "general relativity", "spectral line broadening
"):
- Accept answers that name the same physical
concept / mechanism / object as the reference,
even if the wording differs (e.g. "tidal force"
vs "tidal forces" vs "differential gravitational
 pull from the companion").
- Accept reasonable synonyms and equivalent
technical terms used in the field.
- Reject answers that name a different concept,
even if related (e.g. "radiation pressure" is
NOT equivalent to "tidal forces").
- A correct answer buried inside a longer
explanation is still correct as long as the
named concept is unambiguously the reference one.

- Spelling, capitalization, pluralization and
trivial word-order differences are immaterial.

Do not include any text outside the two required
 sections.

--- Problem ---
{{problem}}

--- Reference solution ---
{{RS}}

--- Reference final answer(s) ---
{{RA}}

--- Student's full solution ---
{{SS}}

--- Answer(s) extracted from student's solution
---
{{SA}}
```

E Application Set redaction from the training material

Because Application Set problems are themselves end-of-chapter exercises drawn from textbooks in $\mathcal{C}$, naively releasing those textbooks as training material would leak every Application Set answer. We therefore physically redact, from the raw markdown that becomes $\mathcal{C}$, both the problem statement and the reference solution of every retained Application Set parent. The process runs in two passes followed by an audit.

Pass 1: problem statements. For each retained Application Set parent, we locate its problem block in its source textbook and replace the block with a single-line marker of the form `<!--[TEST-SETREDACTION]source_problem_id=<id>source=<src>-->`.

Block boundaries are determined by source-specific layout rules, since the eight contributing textbooks use heterogeneous exercise layouts.

Pass 2: solutions and answer keys. A second pass excises the corresponding solutions and answer keys, which for several textbooks live in entirely separate files: Purcell's `electricity_and_magnetism/12_solutions/full.md` aggregates every chapter-end solution; Kleppner ships a parallel `introduction_to_mechanics_solution_manual_by_chapter` directory; each chapter of Morin ends with its own "`X.5Solutions`" section; French's `11_answers/full.md` lists short answers chapter-by-chapter; Taylor & Wheeler's `10a_answers/full.md` concatenates every chapter's answers into a single paragraph, from which we surgically remove only the spans corresponding to redacted parents; Eisberg & Resnick's inline back-of-chapter answer paragraphs are trimmed similarly. For the remaining contributors (Carroll & Ostlie and Palen's *Schaum's Astronomy*), the textbook provides no separate solution section, so Pass 1 alone suffices.

Audit. After both passes, we walk every `full.md` in the corpus, extract 40-character normalised anchors from each retained parent's problem and solution texts, and search for verbatim matches. The released corpus $\mathcal{C}$ contains zero such matches; the same audit additionally confirms that every retained Application Set parent has at least one redaction marker placed in the corpus. The full redaction and audit scripts are released alongside the dataset.

F Convergence curves for the remaining baselines

Figure 2 in the main text plots Bonito's Application-Set sub-problem accuracy against cumulative self-evolution GPU-time. The corresponding curves for the four other baselines are reproduced individually below (Figures 3–6). Compute scales differ by up to $\sim 40\times$ across methods, so each panel uses a method-specific GPU-time axis; the accuracy axis is fixed across all five panels for visual comparison. The qualitative pattern — a fast initial climb, an early plateau, and either no further movement (ACE), noisy oscillation (GEPA), or a late decline

(Intuitor, R-Zero) — is the same across all four methods and consistent with the Bonito panel.

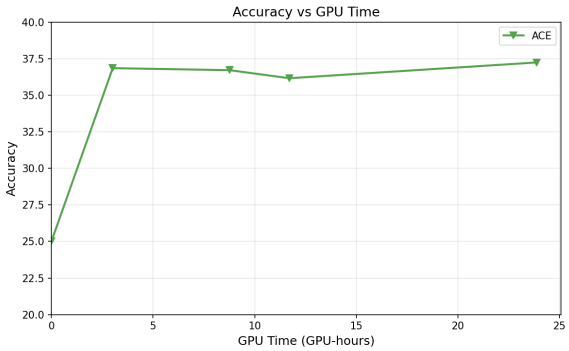

Figure 3: Application-Set sub-problem accuracy of **ACE** against cumulative self-evolution GPU-time. ACE reaches 36.9% within 3 GPU-hr (the first ∼13% of its budget); the remaining ∼21 GPU-hr move accuracy by only +0.4 points.

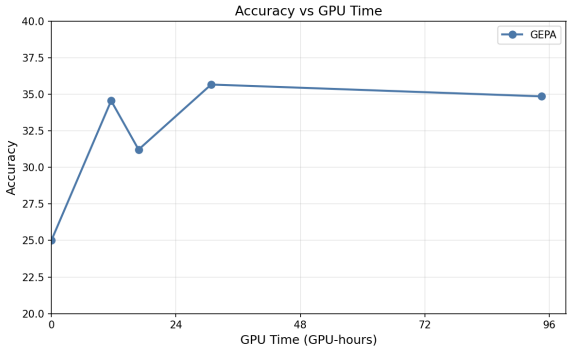

Figure 4: Application-Set sub-problem accuracy of **GEPA** against cumulative self-evolution GPU-time. GEPA hits 34.5% within 12 GPU-hr (∼13% of its budget) and then oscillates in a 4.5-point band; the final 96-hr snapshot (34.8%) is below the 32-hr peak (35.7%).

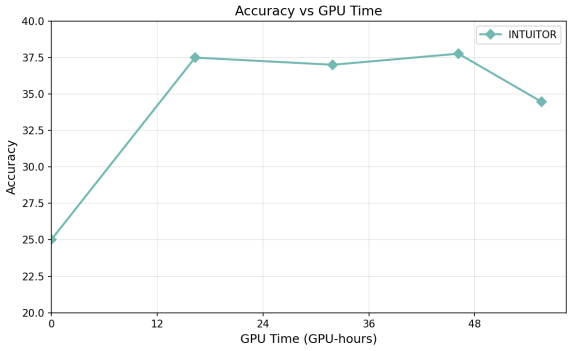

Figure 5: Application-Set sub-problem accuracy of **Intuitor** against cumulative self-evolution GPU-time. Intuitor hits 37.5% within 16 GPU-hr (∼29% of its budget), peaks at 37.8% near 46 GPU-hr, and then drops to 34.5% by the 56-hr endpoint (−3.3 from peak).

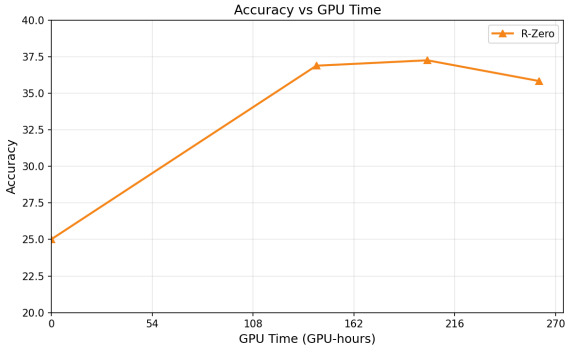

Figure 6: Application-Set sub-problem accuracy of **R-Zero** against cumulative self-evolution GPU-time. R-Zero hits 36.9% near 150 GPU-hr (∼57% of its budget) and peaks at 37.3% near 190 GPU-hr; the final ∼ 75 GPU-hr drop accuracy by 1.5 points to 35.8%.

G Replication detail

We reproduce each baseline from its official repository and adapt it to StudyBench's training-material layers and evaluation protocol. Unless noted otherwise, all changes are implemented by subclassing upstream base classes or adding thin wrapper scripts. Below we summarise the adaptations for each method.

GEPA. GEPA consumes the *Instructions with Answer* layer and evolves a system prompt via reflective genetic search. We made four adaptations:

- **Train/validation split.** We wrote a data loader that shuffles the instruction set and reserves 32 items as a validation set, using the remainder for training. The validation set is kept small because GEPA re-evaluates it after every iteration; a larger set slows each rollout substantially and, in our runs, occasionally caused the evaluation loop to hang for unknown reasons.
- **StudyBench adapter.** We subclass GEPA's `Adapter` class, which is responsible for generation, scoring, and returning textual feedback. Our adapter mirrors the benchmark's multi-turn generation protocol and its rule-plus-LLM verifier, and calls Qwen3-8B to produce the reflective feedback GEPA uses to mutate candidate system prompts.
- **Checkpoint callback.** We register a callback that snapshots the best system prompt seen so far every 100 rollouts. This is a bookkeeping convenience for post-hoc inspection and does not change the search objective or final result.
- **Non-invasive integration.** All of the above are

implemented through inheritance; the upstream GEPA package itself is left unmodified.

Bonito. Bonito consumes the *Corpus* layer to synthesise instruction data and fine-tunes the base model on it. We made three adaptations:

- **Corpus chunking.** We wrote a data loader that splits each textbook in $\mathcal{C}$ into 2,048-token chunks for downstream instruction extraction.
- **Physics-problem task type.** We added a *physics-problem* extraction category to Bonito's task-conditioned generator. A dedicated prompt asks the model to mine standalone physics exercises from textbook passages.
- **Self-evolution loop.** The original Bonito pipeline stops after instruction extraction. We add one further round: instructions produced by Qwen3-8B are fed back to train the same model, closing a single self-evolution cycle on the corpus.

ACE. We make three classes of changes to the open-sourced ACE codebase. *(i) Backbone.* All three roles (Generator, Reflector, Curator) are driven by a single locally-served Qwen3-8B; we strip `<think>` traces between roles, disable thinking on Reflector/Curator (kept on the Generator), and harden the JSON parser against LaTeX escape sequences. *(ii) Playbook management.* We cap the playbook at 12,000 tokens and implement the token-budget Pruning Trigger described in the ACE paper but absent from the released code, which drops the lowest-utility bullets whenever the budget is exceeded; we additionally enable the bulletpoint analyzer for embedding-based deduplication and LLM-driven merging of near-duplicate entries. *(iii) Evaluation.* Rather than use ACE's built-in `eval_only` (whose prompt format, judge, and metric are not aligned with StudyBench), we leave StudyBench's evaluator untouched and inject the playbook into its Generator system message, reducing the ACE-vs-baseline comparison to a single controlled variable.

Intuitor. Intuitor consumes the *Instructions without Answer* layer and trains with GRPO under the policy's own self-certainty as reward. We made two adaptations:

- **StudyBench training split.** We load our own instruction set and partition it into training and validation splits for the GRPO run. Because Intuitor is designed to train without an external reward signal, no verifier is used during optimisation.
- **Validation-time monitoring.** On the held-out validation split we attach the same rule-based verifier used in benchmark evaluation (Section 2.2) to track learning curves and guard against degenerate training trajectories. To keep validation lightweight, we expose only the rule-based component and omit the LLM-judge fallback.

R-Zero. We adapt R-Zero (Huang et al., 2025a) to our setting without altering its core algorithm: we preserve the Challenger/Solver GRPO objectives, the uncertainty reward and BLEU diversity penalty, majority-vote pseudo-labels with $m = 10$ Solver samples per question, the $\hat{p} \in [0.3, 0.8]$ consistency band, and the co-evolution loop, and only rewrite (i) the Challenger and Solver system prompts from math problem setting/solving to physics-olympiad-style problem setting/solving, enforcing a single boxed numerical, symbolic, or equation answer with no units, no proofs, and no plots; (ii) the verifier, replacing `mathruler.grade_answer` with the same rule-based, type-aware judger used by our benchmark (no LLM fallback), with a monkey-patch that routes `sympy.parsing.latex.parse_latex` through `latex2sympy2_extended` to sidestep the `omegaconf/sympy` ANTLR version conflict; and (iii) some pipeline plumbing (local Parquet instead of a HuggingFace round-trip, dynamic checkpoint-path resolution that tolerates mid-training crashes, and `set -e` propagation across stages). We use `Qwen3-8B` (Instruct) as the base model and run 3 co-evolution iterations (the saturating regime reported in the original paper), with 6 Challenger and 20 Solver GRPO steps per iteration and `max_response_length`=8192, on a single 8×A800-80GB node; one full run takes approximately ten days of wall time. Final evaluation uses our benchmark's standard protocol, not R-Zero's built-in evaluation.

H LLM Usage

During preparation of this manuscript we used large language models (LLMs) as editorial and data-processing tools to support the writing and dataset preparation workflow. Their assistance included PDF-to-Markdown conversion and automated correction of common OCR and formatting errors. Also we use it with editorial help on the manuscript text such as grammar and style polishing, sentence rephrasing, paragraph restructuring, and drafting initial versions of descriptive passages.

Crucially, all LLM-assisted outputs were treated as drafts, every suggestion was reviewed, edited, and approved by the authors before incorporation. We disclose this use here and in the submission form in accordance with community guidance.