

Fine-tuning, Reinforcement Learning and Self-Learning

**NASA AI/ML STIG seminar
October 19th 2026**



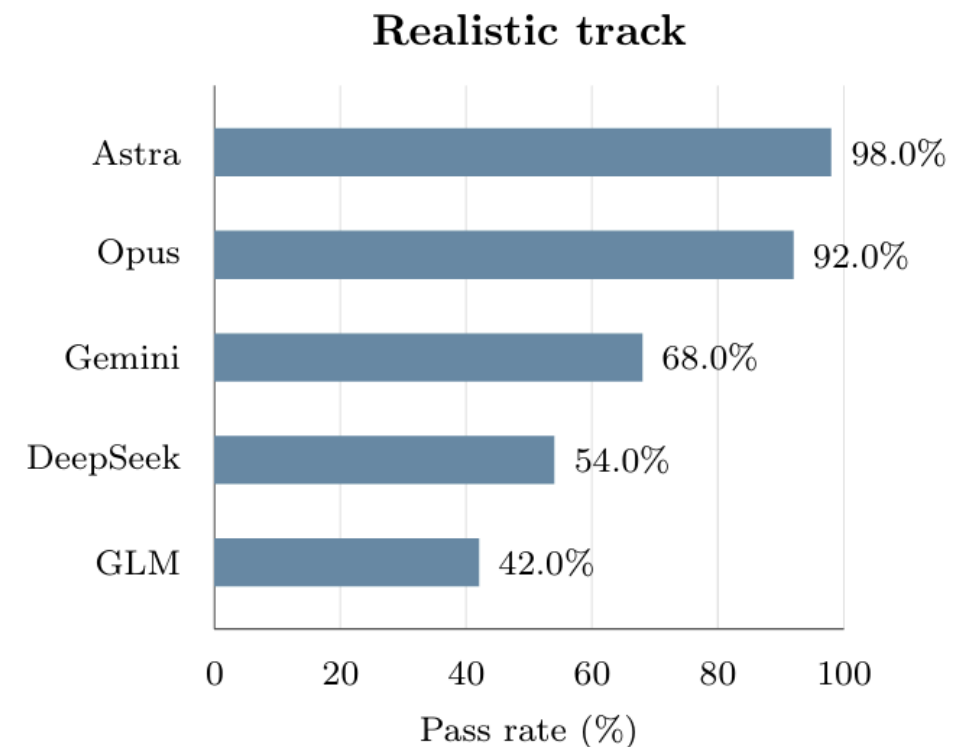
Moritz Münchmeyer, University of Wisconsin-Madison

Recent progress in AI for physics

- Theoretical physics benchmarks are approaching saturation.

TheoPhysEval: Assessing AI agents at autonomous theoretical physics problem solving

2610.16733



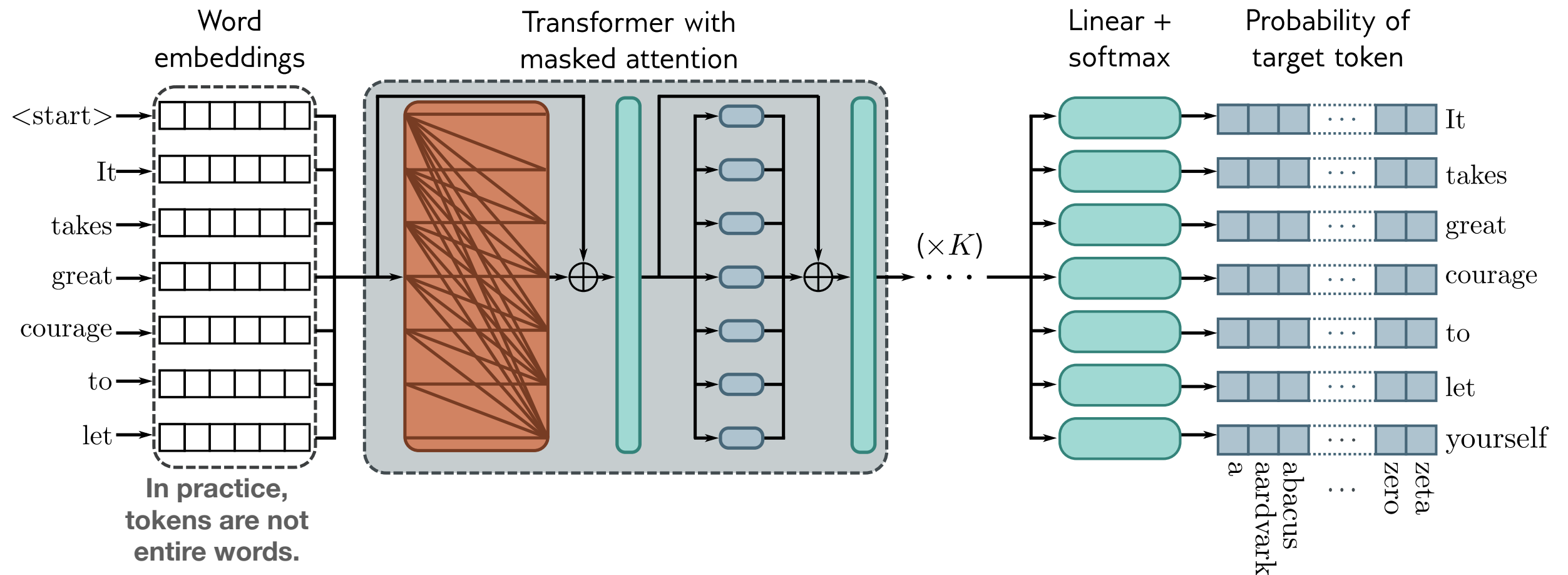
- Many **recent papers use AI for substantial mathematical calculations.**
- Autonomous discovery of important physics results is not yet common. A central challenge is **choosing meaningful, well-defined research problems.**
- Given the importance AI systems, it is **important to understand how they are being trained and how they work.** Thus this lecture series!

Stages of LLM training

From pre-training to RL

Recall: Pretraining LLMs

- LLMs learn a probability distribution for the next token using an autoregressive transformer.



- Pretraining uses large corpora of text, code and other data, including scientific papers.

- Next token prediction loss:
$$\mathcal{L}(\theta) = - \sum_{t=1}^T \log p_{\theta}(y_t | y_{<t})$$

Stages of training

- **Pretraining:** next-token prediction on broad text, code and other data builds general capabilities.
- **Midtraining:** continue the same objective on a more targeted data mixture.
- **Supervised fine-tuning (SFT):** imitate curated prompt–response examples, including reasoning traces. Usually only response tokens contribute to the loss.
- **RLVR:** sample solutions, verify outcomes automatically and reinforce successful attempts. GRPO is one training algorithm.
- **RLHF / RLAI:** use human or AI preferences to optimize response quality, including tasks without exact answers.
- These stages can repeat or overlap. **Tool use and agent training** can combine SFT and RL.

Supervised Fine-Tuning

Fine-tuning with SFT

Supervised fine-tuning (SFT)

Training data pairs a prompt x , specifying the problem, with an expert solution y^* .

An expert solution is a sequence of tokens:

$$y^* = (y_1^*, \dots, y_T^*)$$

The LLM generates the answer auto-regressively:

$$\pi_\theta(y^* \mid x) = \prod_{t=1}^T \pi_\theta(y_t^* \mid x, y_{<t}^*),$$

Here, π_θ is the LLM model (or “policy”), and the preceding expert tokens are

$$y_{<t}^* = (y_1^*, \dots, y_{t-1}^*).$$

The SFT loss is the negative log-likelihood of the expert solution:

$$\mathcal{L}_{\text{SFT}} = -\log \pi_\theta(y^* \mid x) = -\sum_{t=1}^T \log \pi_\theta(y_t^* \mid x, y_{<t}^*).$$

Training makes the expert solution a likely answer to the prompt, token by token.

Example SFT data

x — prompt:

A mass m is attached to a spring with constant k .

Find its angular frequency.

y — expert response:

**Newton’s law gives
 $m \ddot{q} = -kq$.**

Compare with $\ddot{q} = -\omega^2 q$:

$\omega = \sqrt{(k/m)}$.

Software stack for post-training

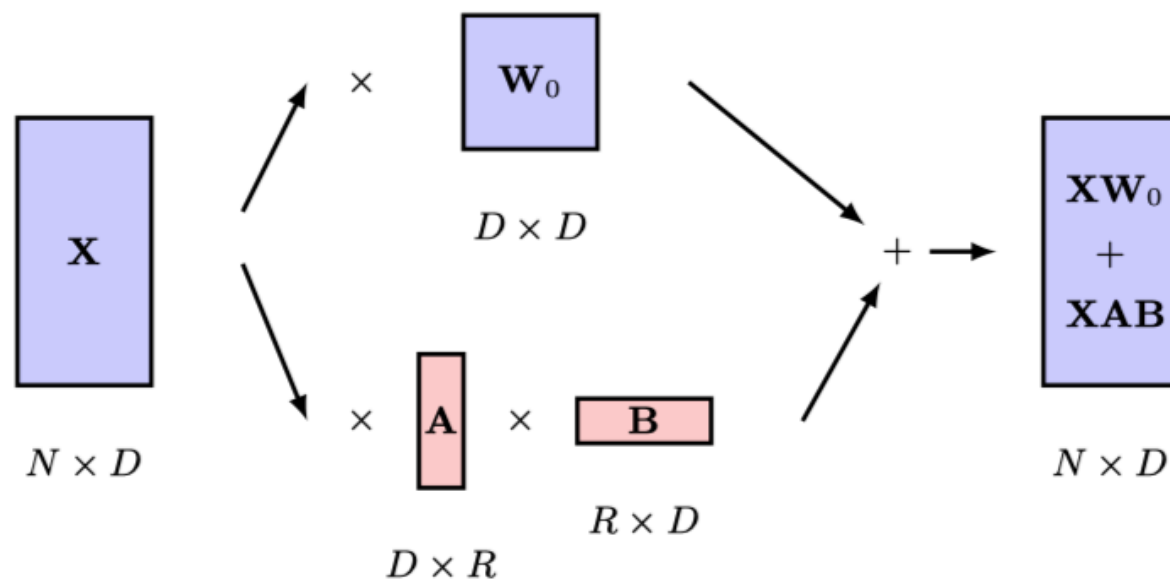
- Mature libraries provide optimized building blocks for post-training.
- **SFT:** Hugging Face TRL (SFTTrainer), NVIDIA NeMo and Axolotl provide established training workflows.
- **verl:** distributed RL training, with flexible model and rollout backends for research workflows.
- **NeMo RL:** NVIDIA's framework for distributed RL, with support for large GPU training jobs.
- Choose a stack based on the model, hardware and rollout engine, then profile memory and throughput.

Memory requirements (Example)

- **7B example:** 10K tokens total (prompt + response), 16-bit weights and full AdamW training.
- **Inference:** $\approx 22\text{--}24$ GB total for one sequence, including ≈ 14 GB weights and ≈ 5 GB KV cache.
- **SFT:** budget $\approx 150\text{--}170$ GB total, including ≈ 126 GB model states. Assume microbatch 1, FlashAttention and layer activation checkpointing.
- **RLVR / GRPO with KL:** budget $\approx 180\text{--}200$ GB total for 4 concurrent 10K-token rollouts, a frozen reference policy and microbatch 1 updates. Same training optimizations as SFT.
- These are illustrative estimates across GPUs, using a **32-layer model with standard multi-head attention**. Actual peaks vary by implementation.

SFT with LoRA

- **Low-rank adaptation (LoRA)** freezes pretrained weights and trains small low-rank updates to selected matrices (Hu et al., 2021).
- Advantage: Needs less much GPU memory, potentially less training data for fine-tuning (regularization).



Schematic illustration low-rank adaptation showing a weight matrix W_0 from one of the attention layers in a pre-trained transformer. Additional weights given by matrices A and B are adapted during fine-tuning and their product AB is then added to the original matrix for subsequent inference.

Example: rank 16

W_0 : 4096×4096
16.8M frozen parameters

A : 4096×16
 B : 16×4096

131,072 trainable parameters
 $\approx 0.8\%$ of the full matrix

Typical ranks: 8–64

Reinforcement Learning

Fine-Tuning with RLVR

Fine-tuning reasoning with RLVR

RLVR: reinforcement learning with verifiable rewards.

Sample: generate K candidate solutions for a prompt x using the “policy” given by the LLM.

$$y_i \sim \pi_{\text{old}}(\cdot \mid x), \quad i = 1, \dots, K$$

Verify: check the final answer or run code tests.

$$r_i = r(x, y_i) = \begin{cases} 1 & \text{passes the verifier} \\ 0 & \text{otherwise} \end{cases}$$

Update: reinforce successful solutions, e.g. with GRPO.

The **RL update needs no expert reasoning trace**.
Solutions can differ from human demonstrations.

x — prompt

Spring with mass attached.
 $m = 1 \text{ kg}$, $k = 4 \text{ N/m}$.
What is ω ?

y_1 — sampled attempt

$m \ddot{q} = -4q$, so $\omega^2 = 4$.
 $\omega = 2 \text{ rad/s}$.

Reward 1 → reinforce

y_2 — sampled attempt

$\omega = k/m = 4 \text{ rad/s}$.

Reward 0 → suppress

GRPO algorithm

- **Before GRPO:** PPO-based RLHF (e.g. InstructGPT) trained an additional value model (“critic”) to estimate advantages, adding memory and training cost.
- **Innovation:** Group Relative Policy Optimization (DeepSeekMath, 2024) compares several answers to the same prompt. Normalized group rewards replace the learned critic, reducing overhead while retaining PPO-style clipping.
- **Use by Chinese labs:** DeepSeek used GRPO for R1. Alibaba’s Qwen team also used GRPO, then introduced GSPO for later Qwen3 models.
- **Use by US labs:** NVIDIA reports GRPO training for Nemotron. Public reports do not establish whether OpenAI or Anthropic use GRPO in their flagship models.

GRPO: full objective

For one prompt x , sample K trajectories from the rollout policy.

$$y_i \sim \pi_{\text{old}}(\cdot \mid x), \quad i = 1, \dots, K$$

Minimize the average loss over trajectories and their tokens:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{K} \sum_{i=1}^K \frac{1}{T_i} \sum_{t=1}^{T_i} [\min(\overset{\substack{\rho_{i,t}: \text{ policy ratio} \\ \downarrow}}{\rho_{i,t}}} A_i, \text{clip}(\rho_{i,t}, 1 - \varepsilon, 1 + \varepsilon) A_i) - \beta D_{i,t}(\theta)]$$

K trajectories
 T_i tokens in trajectory i

A_i : group-relative
advantage

ε : clipping threshold
Limits the incentive for
large policy updates

β : KL strength
 D_i : KL divergence
Penalty for
departing from the
reference policy

GRPO: policy ratio and KL

Token context: prompt and preceding generated tokens

$$h_{i,t} = (x, y_{i,<t})$$

Policy ratio: current probability / rollout probability.

$$\rho_{i,t}(\theta) = \frac{\pi_{\theta}(y_{i,t} \mid h_{i,t})}{\pi_{\text{old}}(y_{i,t} \mid h_{i,t})}$$

We can do multiple gradient updates on the same rollouts (for the old policy). Clip to limit each round of updates.

KL divergence: departure from a frozen reference policy

$$D_{i,t}(\theta) = D_{\text{KL}}(\pi_{\theta}(\cdot \mid h_{i,t}) \parallel \pi_{\text{ref}}(\cdot \mid h_{i,t}))$$

The reference is typically the SFT model. The KL penalty discourages large changes from that starting policy. Often omitted.

GRPO: binary rewards

A verifier assigns $r_i = 1$ for a correct answer and $r_i = 0$ otherwise.

Within this group, the success fraction and reward spread are

$$p = \frac{1}{K} \sum_{i=1}^K r_i, \quad s = \sqrt{\frac{1}{K} \sum_{i=1}^K (r_i - p)^2} = \sqrt{p(1-p)}$$

Every token in a trajectory receives the same “group relative” advantage:

$$A_i = \frac{r_i - p}{s}, \quad 0 < p < 1$$

$$w_+ = \frac{1-p}{s} > 0 \quad (\text{correct, } r_i = 1)$$

$$w_- = -\frac{p}{s} < 0 \quad (\text{incorrect, } r_i = 0)$$

Correct answers have positive advantage. Incorrect answers have negative advantage.

All pass or all fail: $A_i = 0$

GRPO: trajectory weights

Remove clipping and set $\beta = 0$. The loss gradient simplifies to:

$$\nabla_{\theta} \mathcal{L}_{\text{unclipped}}(\theta) = -\frac{1}{K} \sum_{i=1}^K \frac{A_i}{T_i} \sum_{t=1}^{T_i} \rho_{i,t} \nabla_{\theta} \log \pi_{\theta}(y_{i,t} \mid h_{i,t})$$

At the rollout policy, every token ratio is 1 (first gradient step on the rollouts).

$$\begin{aligned} \nabla_{\theta} \mathcal{L}_{\text{GRPO}}(\theta) = & -\frac{1}{K} [w_+ \sum_{i:r_i=1} \frac{1}{T_i} \sum_{t=1}^{T_i} \nabla_{\theta} \log \pi_{\theta}(y_{i,t} \mid h_{i,t}) && \text{upweight correct rollouts} \\ & + w_- \sum_{j:r_j=0} \frac{1}{T_j} \sum_{t=1}^{T_j} \nabla_{\theta} \log \pi_{\theta}(y_{j,t} \mid h_{j,t})] && \text{downweight wrong rollouts} \end{aligned}$$

Gradient descent reinforces correct trajectories and suppresses incorrect ones, averaging over their tokens.

Note: If a rollout has even one small mistake that affects the outcome, all of it gets down-weighted. This seems like a very inefficient learning algorithm, but it works very well.

Fine-tuning models for theoretical physics

arXiv:2604.18936,

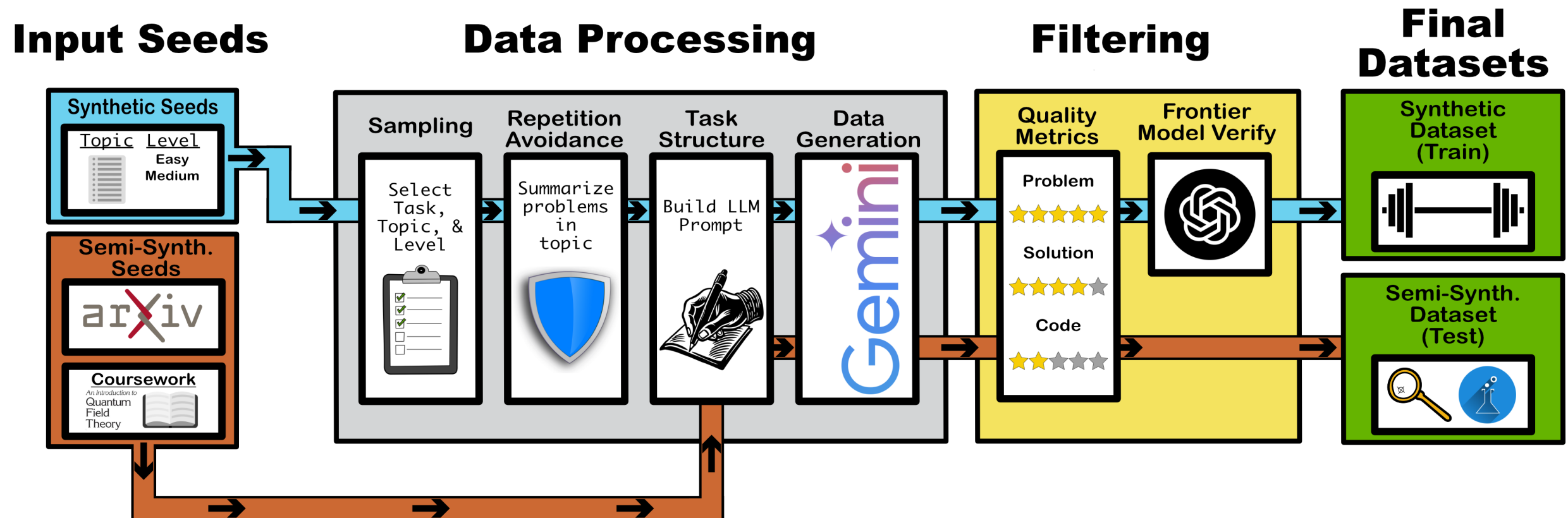
Published in Machine Learning: Science and Technology

Led by Nate Woodward

Can we do RL specifically for theoretical physics?

- Small open models let us **inspect training dynamics and generated reasoning traces**.
- **Academic RL experiments** can test alternative objectives and rewards, including exploration, reliability and self-play.
- University compute limits model size, but still permits controlled experiments in new domains and with new tools.
- **Our goal** is to understand how RL changes physics reasoning and which training choices improve it.

Dataset generation pipeline



- Frontier teacher models help generate synthetic and semi-synthetic QFT datasets.
- **Result:** about 2,000 automatically verifiable problems spanning several difficulty levels.

Our QFT problem set for RLVR

- Simple verifiable problems suitable for small models.
- Get the data set on huggingface!

Example problem:

Consider a complex scalar field Φ in four spacetime dimensions described by the Lagrangian:

$$\mathcal{L} = (\partial_\mu \Phi^\dagger)(\partial^\mu \Phi) - V(\Phi) \quad (15)$$

with the potential

$$V(\Phi) = \lambda \left(|\Phi|^2 - \frac{v^2}{2} \right)^2 \quad (16)$$

where $\lambda > 0$ and $v > 0$ are real parameters. This theory has a global $U(1)$ symmetry, $\Phi \rightarrow e^{i\alpha} \Phi$. The potential induces spontaneous symmetry breaking (SSB). We choose the vacuum expectation value (VEV) to be real: $\langle \Phi \rangle = \frac{v}{\sqrt{2}}$. The quantum fluctuations around this VEV are parameterized by real scalar fields $h(x)$ (the massive mode) and $\pi(x)$ (the massless Goldstone boson) as:

$$\Phi(x) = \frac{1}{\sqrt{2}}(v + h(x) + i\pi(x)) \quad (17)$$

The conserved Noether current associated with the $U(1)$ symmetry is given by:

$$J^\mu(x) = i \left((\partial^\mu \Phi^\dagger) \Phi - \Phi^\dagger (\partial^\mu \Phi) \right) \quad (18)$$

According to Goldstone's theorem, the massless mode $\pi(x)$ is created from the vacuum by the action of this current. The matrix element of the current between the vacuum $|0\rangle$ and a one-Goldstone boson state $|\pi(p)\rangle$ with momentum p is parameterized by a constant F , known as the decay constant:

$$\langle 0 | J^\mu(x) | \pi(p) \rangle = -iF p^\mu e^{-ip \cdot x} \quad (19)$$

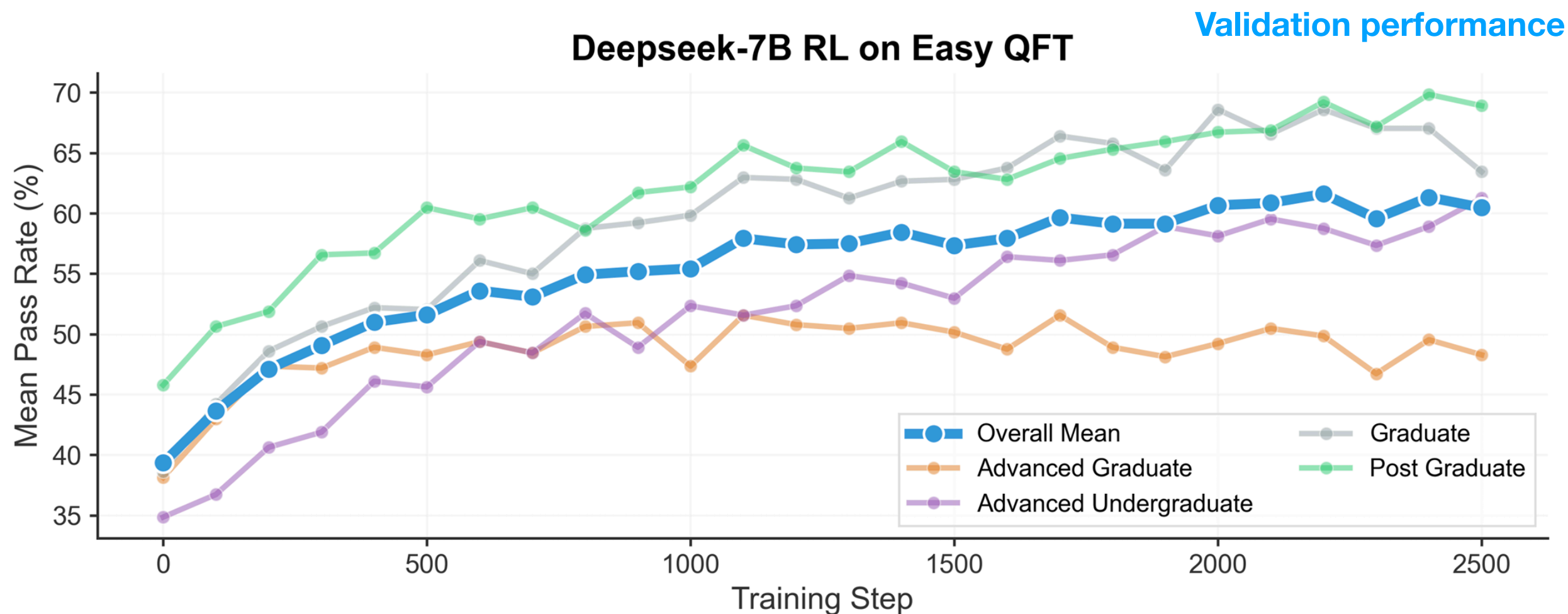
Task: By expanding the current $J^\mu(x)$ in terms of the fields $h(x)$ and $\pi(x)$, determine the decay constant F . Find the numerical coefficient C in the relation $F = C \cdot v$.

Conventions: The one-particle states $|\pi(p)\rangle$ are normalized such that the matrix element of the field operator is $\langle 0 | \pi(x) | \pi(p) \rangle = e^{-ip \cdot x}$.

<https://huggingface.co/datasets/nswoodward/VerifiableQFT>

Results on RL fine-tuning

- We fine-tuned small models (Qwen 7B) with GRPO on easy QFT problems.
- Compute: four H200 GPUs for about two weeks, provided by Perimeter Institute.



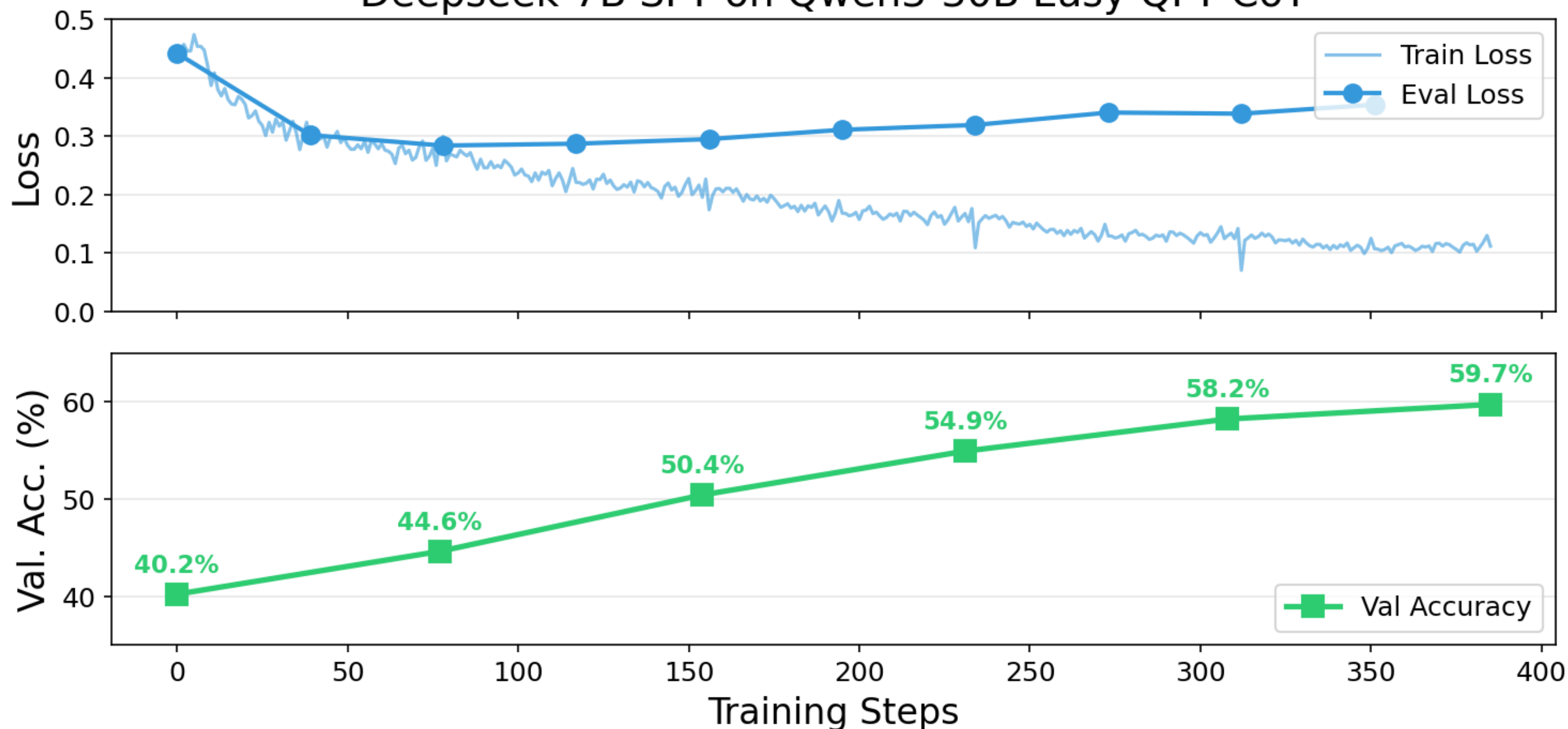
Setup: DeepSeek-R1-Distill-Qwen-7B, GRPO, 32 rollouts per prompt, no KL penalty, Clip-Higher

Rollouts: $4 \text{ prompts/step} \times 32 \text{ rollouts/prompt} \times 2,500 \text{ steps} \approx 320,000$.

Results on SFT fine-tuning

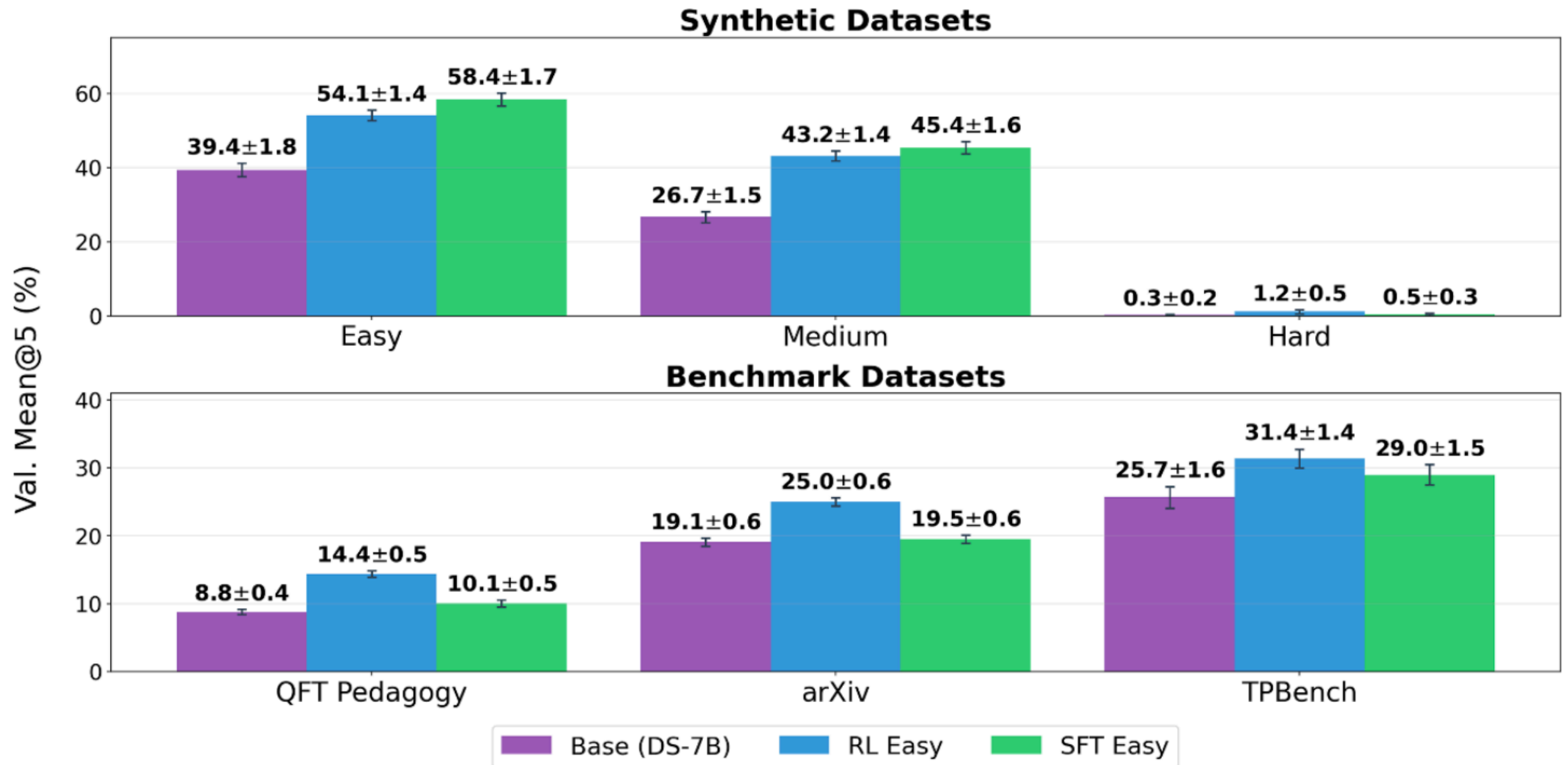
- We used a Qwen3-30B model as a teacher to provide chain-of-thoughts.
- Validation accuracy on **7B model improves from 40.2% to 59.7%**.

Deepseek-7B SFT on Qwen3-30B Easy QFT CoT



SFT vs RL

SFT better in-distribution



RL generalizes better

Error classification

✖ Factual Error

Problem 899, CoT Step 5

Model claim: “ $(\gamma^5)^2 = -\mathbb{I}$ ”

Correct fact: γ^5 squares to the identity, $(\gamma^5)^2 = \mathbb{I}$.

Misrecalled property of the chirality matrix propagates through the subsequent projection operator algebra.

🧮 Mathematical Error

Problem 993, CoT Step 4

Model writes: $(2x - 1)^2 = x^2 - 2x + 1$

Correct expansion: $(2x - 1)^2 = 4x^2 - 4x + 1$.

The factor of 4 is dropped from the leading coefficient, producing incorrect polynomial coefficients for the Feynman parameter integral.

⚠ Logical Error

Problem 49, CoT Step 4

Model derives: Amplitude scales as $\mathcal{A}(z) \sim z^2$ at large z .

Model concludes: “...therefore the amplitude scales as z^{-2} .”

The derivation is correct, but the conclusion contradicts it—inverting the sign of the exponent and reversing the large- z behavior.

</> Executional Error

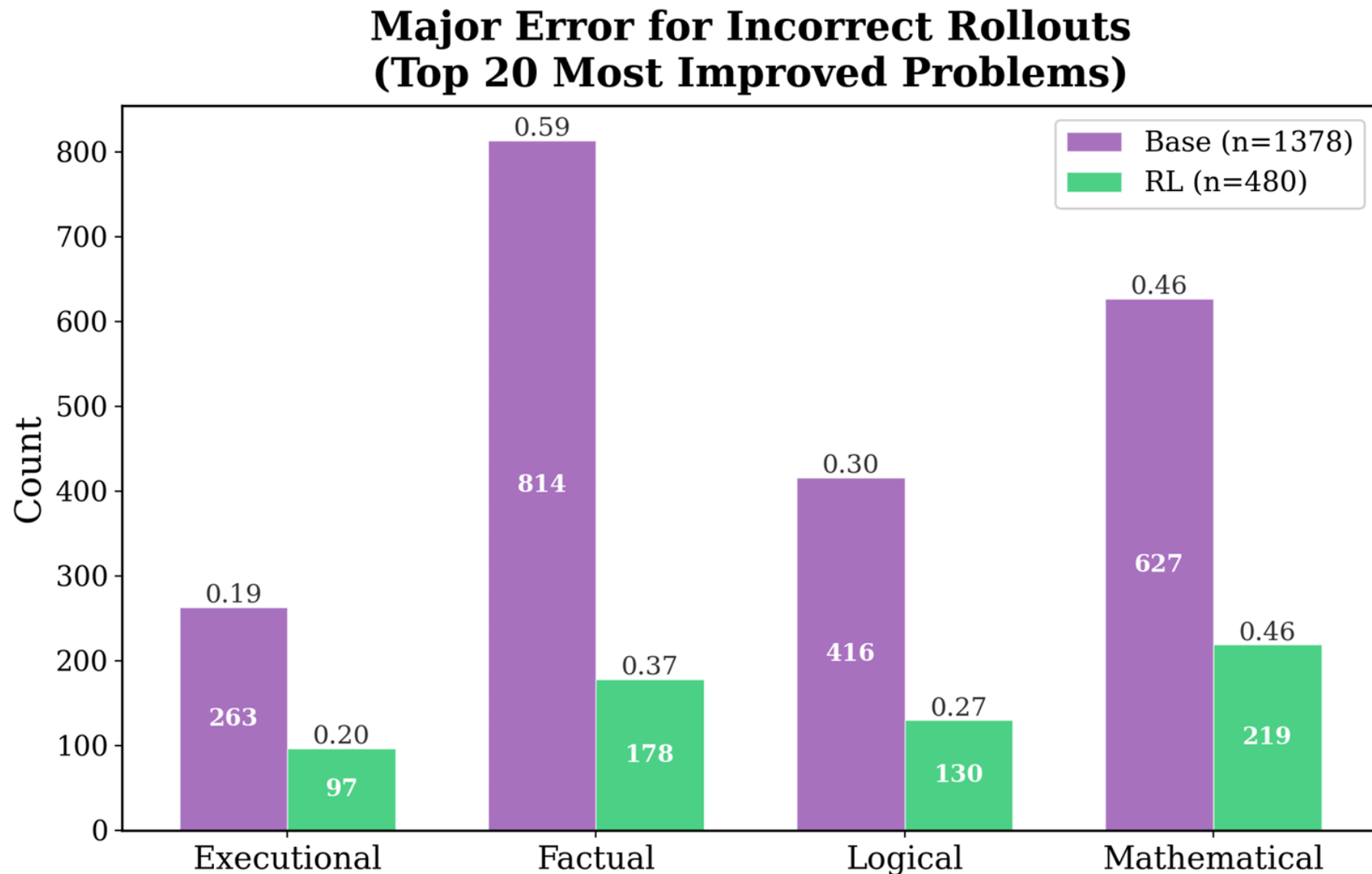
Problem 1199, CoT Step 8

Derivation: Correctly obtains the BPS energy coefficient $C = \frac{1}{4}$.

Code: `C = 1/3` ← **bug**

The analytical reasoning is correct, but the implementation assigns a different numerical value, a transcription error visible only in the code.

Error analysis



- **LLM-based error classification:** gpt-oss-120b and Sonnet 4.6.
- Among the 20 most improved problems, factual errors show the largest reduction.

Lessons from our experiments

- **RL convergence is difficult.** Across several starting models and hyperparameter settings, many of our runs failed.
- **Generated reasoning traces** are often verbose or repetitive. They should not be treated as faithful accounts of human-like reasoning.
- **Pass@1 improves more than pass@32** in our experiments. Pass@k measures whether at least one of k attempts succeeds.
- This echoes **Yue et al. (2025)**, “**Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?**” arXiv:2504.13837.

Self-Learning

Can models learn without
more human data?

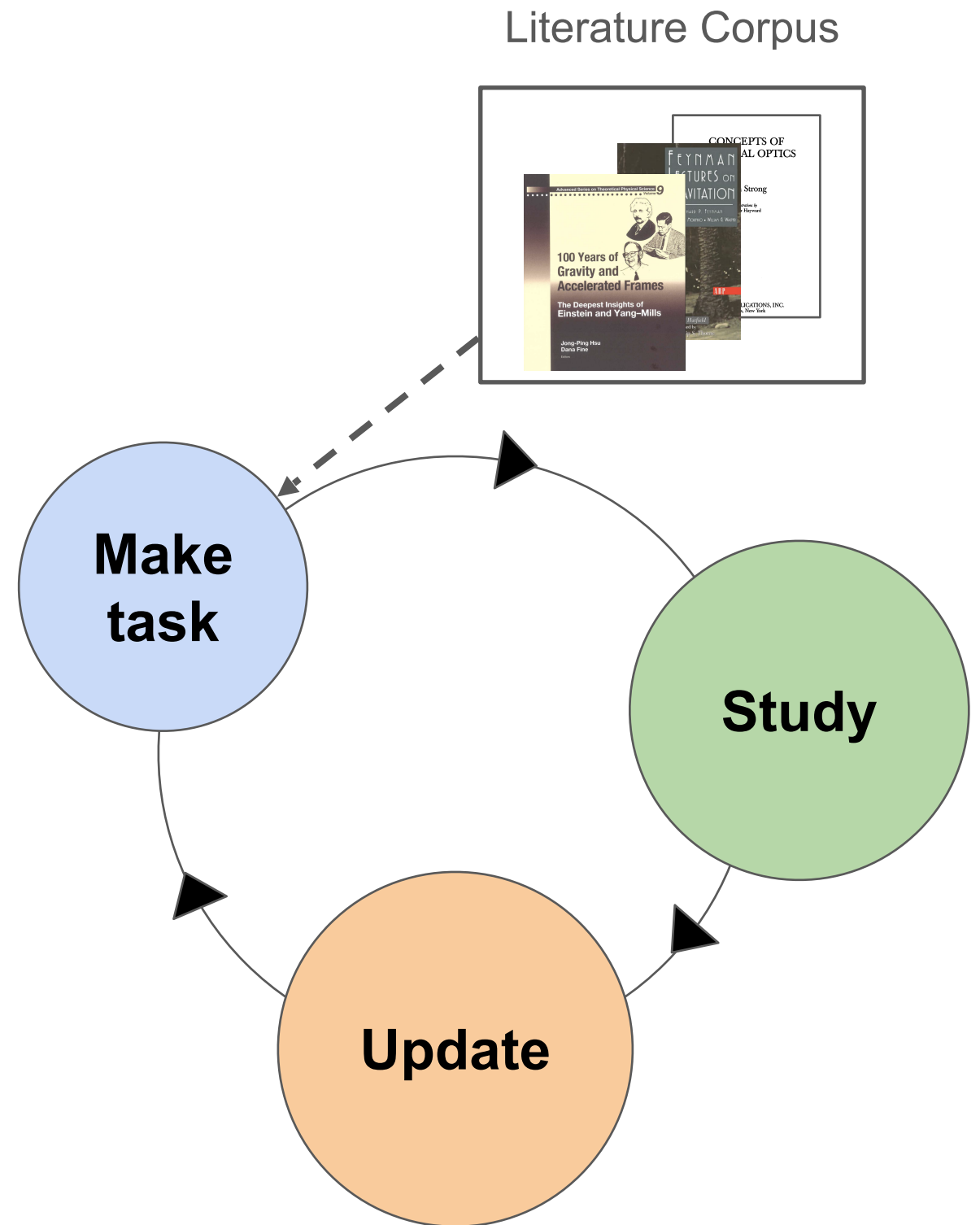
Led by Nate Woodward

Why self-learning?

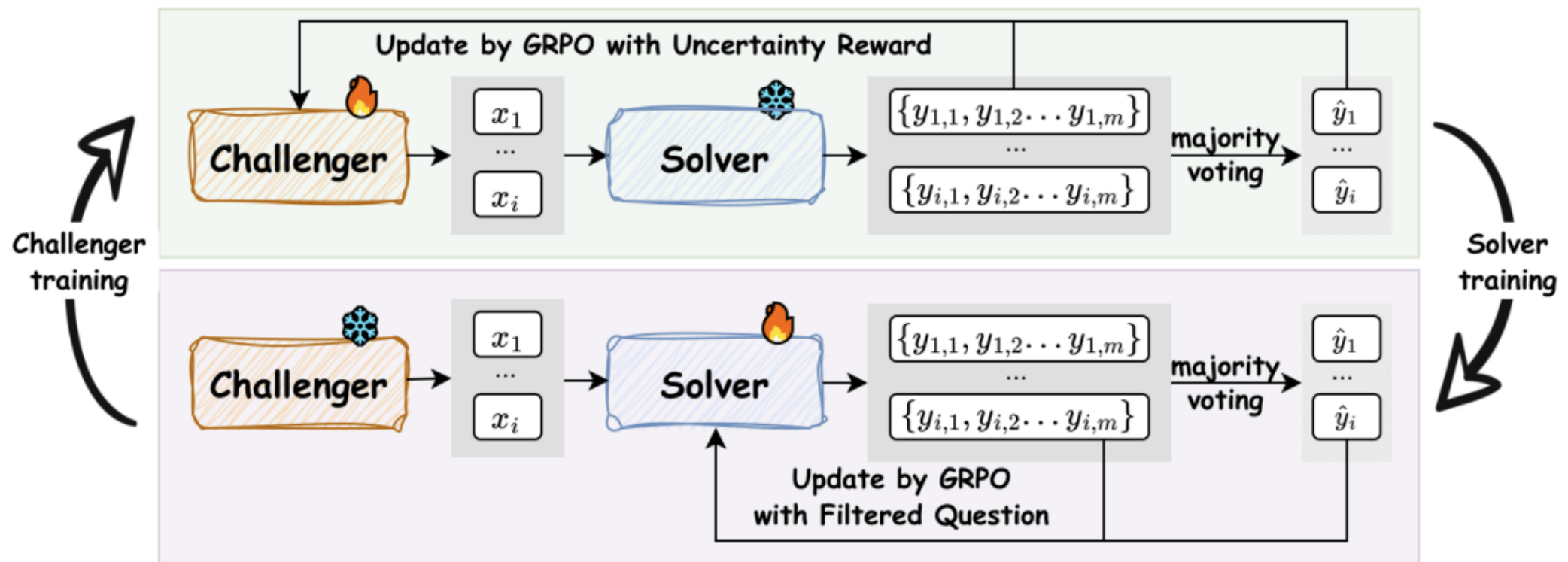
- **High-quality scientific reasoning data** is scarce and expensive to curate. Data is likely the single most important driver of the progress in recent years.
- **Whether or not models can improve significantly without human training data** (e.g. without RL problems with known ground truth) is **critical to AGI timelines**.
- **Academic self-learning results remain mixed**. Compute budgets, starting checkpoints and the learning method all affect the outcome.
 - Many studies start from base models. Improving an already strong reasoning model is a harder test.
- **Here we focus** on models generating training tasks and learning from their attempts, rather than redesigning their own architecture or training code (**“Recursive Self-Improvement” RSI**).

Make your own training data

- One model serves as both challenger and reasoner.
- **Challenger:** creates tasks near the model's learning frontier, optionally using a document.
- **Reasoner:** attempts the generated tasks, with or without access to study material.
- **Learning signal:** a verifier or pseudo-label supplies feedback for the update.
- **Goal:** keep generating tasks that are neither trivial nor consistently unsolvable.



Verification: pseudo-labels

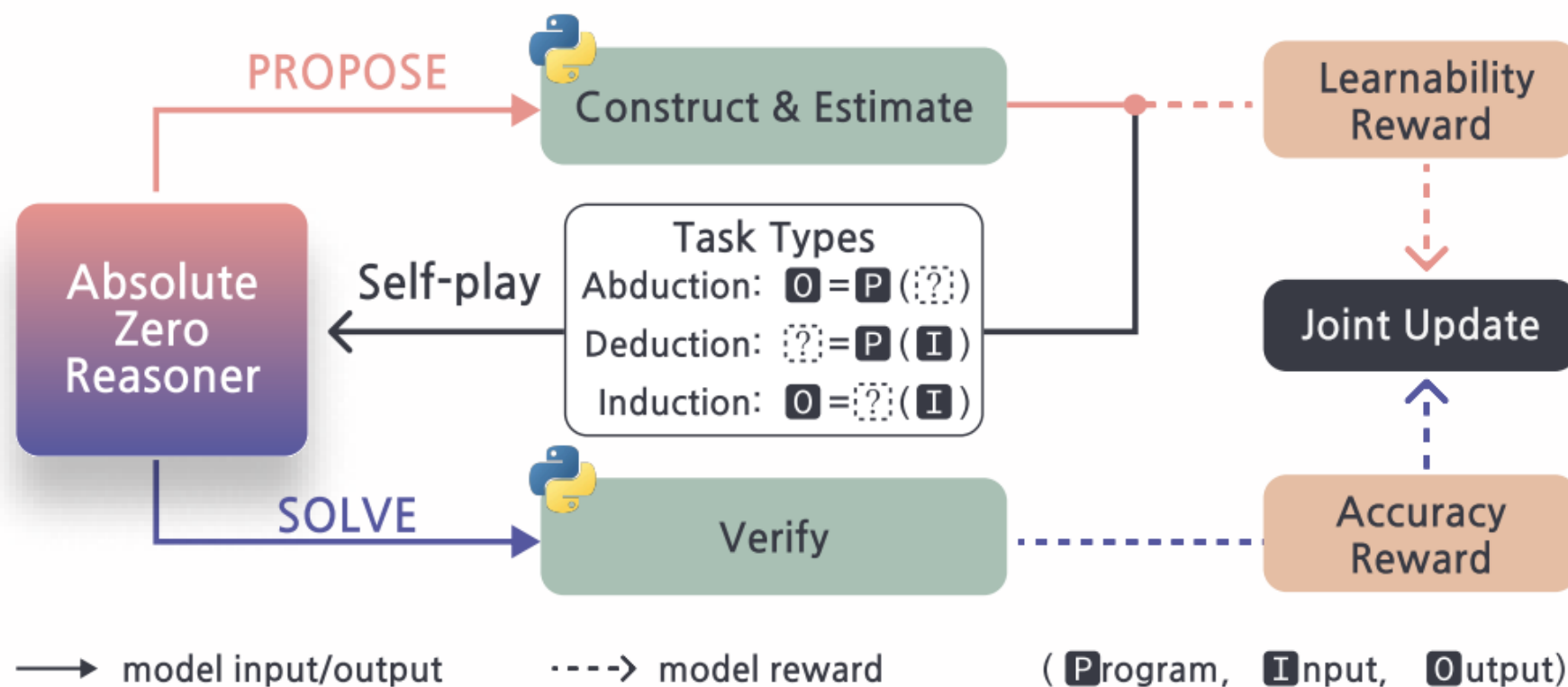


Huang, C. et al Aug 2025

- **R-zero (2508.05004)** does self-learning with zero training data.
- **Verification:** solver majority vote produces a pseudo-label, with no guarantee of correctness.
- **Training:** Reward the solver for matching that pseudo-label, as in RLVR. Reward challenger to find questions at the learning frontier (50% accuracy).
- However, this method degrades after a few iterations.

Verification: external environment

- **Absolute Zero (Zhao et al., 2025, 2505.03335)** uses a **code executor** as the verifier.
- The proposer creates programs and associated reasoning tasks. The solver predicts a missing input, output or program.
- Execution supplies a checkable outcome, reducing reliance on model-generated correctness judgments.



Example

Given the following Python function:

```
def f(nums):  
    s = 0  
    for n in nums:  
        if n % 2 == 0:  
            s += n * n  
        else:  
            s -= n  
    return s
```

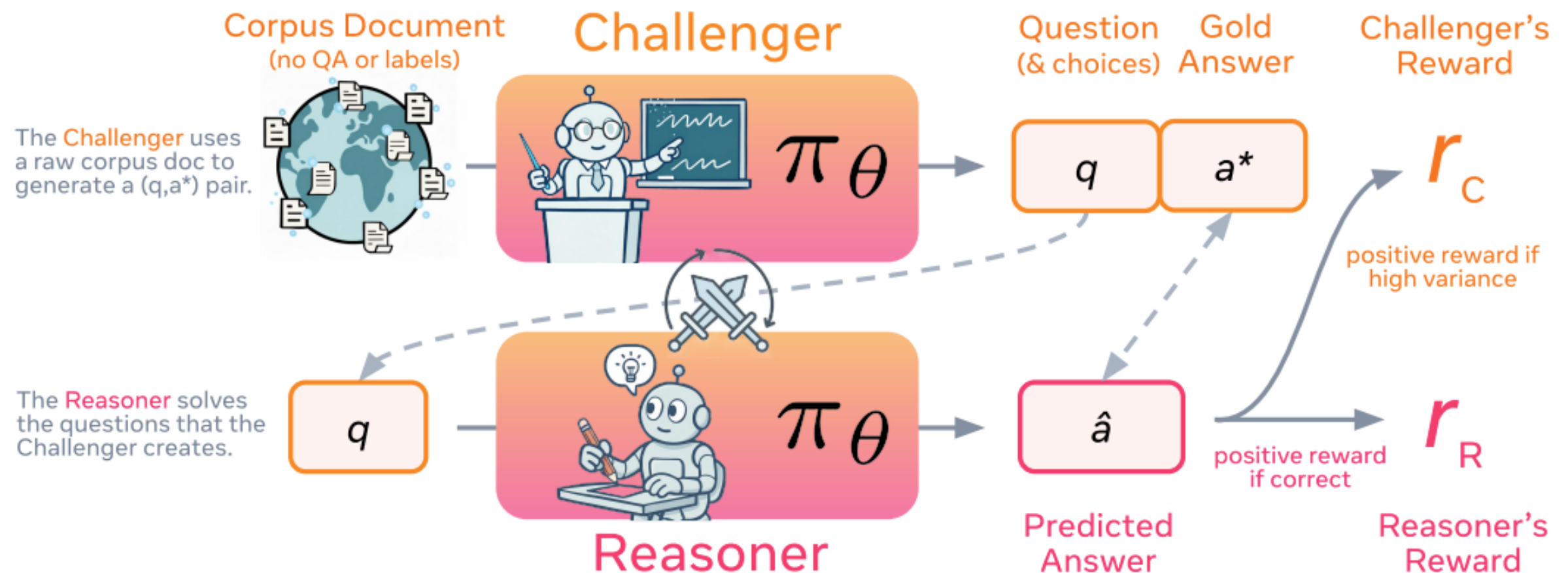
What is the output of `f([1, 2, 3, 4])`?

Can LLMs self-correct?

- **Early studies** found that **models struggle to detect and correct their own reasoning errors** without external feedback.
 - Example: Huang et al. (2023), “Large Language Models Cannot Self-Correct Reasoning Yet,” arXiv:2310.01798.
- In addition, **novelty is hard to guarantee because a problem or solution may have appeared in pretraining data.**
 - Older checkpoints (or “vintage LLMs”) help test contamination, but have weaker capabilities.
- Recent AI-generated mathematics changes the picture somewhat. Models are now able to **solve manifestly novel math problems and self-verify** with good accuracy (even before LEAN formalization).

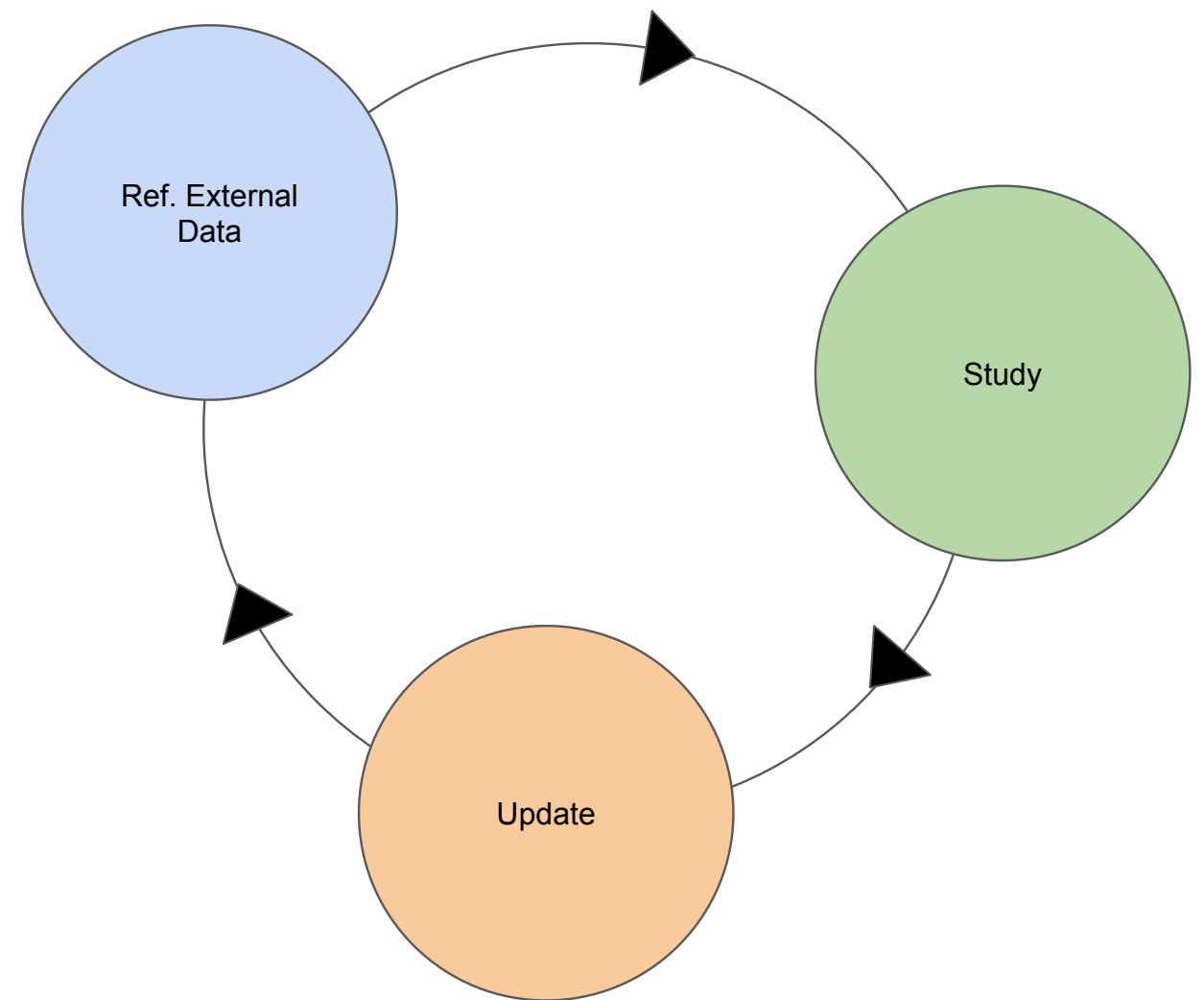
Self-play with literature data

- **SPICE** gives the challenger **access to a corpus document (not a set of problems) to construct reasoning tasks. Close to how humans study.**
- For science, textbooks and papers can supply grounded practice problems at the reasoner's learning frontier.



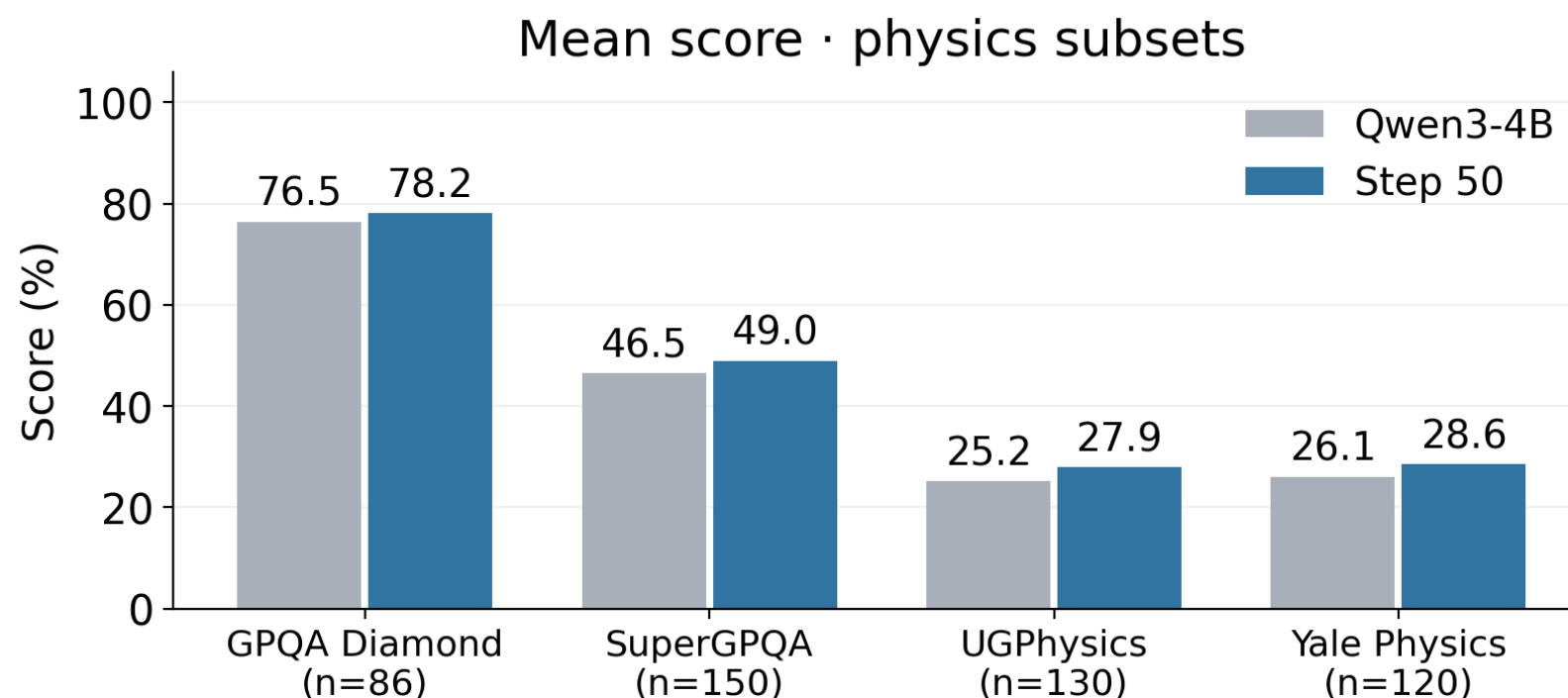
Our project: Teaching yourself physics

- Give a model a scientific corpus, useful study methods and time to practice.
- Let it generate tasks, attempt solutions and update from feedback.
- **Can this loop** improve scientific reasoning in a single model within a closed system?



Self-play experiments: preliminary results

- Qwen3-4B self-play on physics textbooks. **Preliminary results.**
- **Tasks:** free-response and multiple-choice questions.
- **Generation:** up to 40K response tokens, temperature 1.2.
- **Learnability:** about 6% of generated questions were useful for RL, neither too easy nor too hard.
- **Compute:** about 8 hours on one H200 node at FirstPrinciples.
- **Corpus:** 13K textbook sections, about 1K tokens each.



Subsampled validation; GPQA uses all 86 physics questions.

Speculations on self-learning

- Several **problems of existing LLMs appear to be connected:**
 - sample efficiency, robustness, continual learning and even alignment
- SFT and RL use **global gradient updates**. They may be the only part of an effective system for continual learning.
- **Human learning combines multiple forms of memory with signals such as curiosity and surprise.**
- The difficulty of self-learning suggests that **different learning mechanisms may be needed**. These could still be built around existing LLM architectures.

Backup