CS 639: Foundation Models
RLVR

Fred Sala

University of Wisconsin-Madison

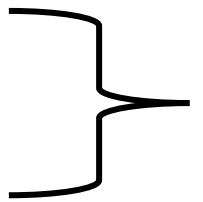
March 26, 2026

Announcements

- **Homework 4**: after break
- **Project---form groups!**:
  - Information here:
  - https://pages.cs.wisc.edu/~fredsala/cs639/files/project_info_639.pdf

- Class outline (prior to break)

Thursday March 26	Reasoning II: RLVR



Improving FMs

Outline

- **Verifiers**
  - Notion of “verifiers”, challenges for oracle verifiers, alternatives & weak verifiers, test-time scaling, ORMs and PRMs
- **RL with Verifiers**
  - Objectives, more on PPO, group-relative policy optimization (GRPO), reward construction, applications

Outline

- **Verifiers**
  - Notion of “verifiers”, challenges for oracle verifiers, alternatives & weak verifiers, test-time scaling, ORMs and PRMs
- **RL with Verifiers**
  - Objectives, more on PPO, group-relative policy optimization (GRPO), reward construction, applications

RL Outside of Alignment

- Let's get back to building **a good model**-doesn't need to be within the context of alignment
  - This means we don't have human preference data, but potentially something else

Where does RL fit in here?

- And what are the new reward models going to look like?
- One simple approach: "rewards" for just the correct answers
  - But, unlike in the supervised case, not just one solution

Verifiers

- Suppose we have a way of determining an answer is correct
  - Note: doesn't mean a full generation has to be identical to a ground-truth answer.

This is especially useful for e.g., math problems

Problem 12

The set of points in 3-dimensional coordinate space that lie in the plane $x+y+z=75$ whose coordinates satisfy the inequalities

$$x-yz<y-zx<z-xy$$

forms three disjoint convex regions. Exactly one of those regions has finite area. The area of this finite region can be expressed in the form $a\sqrt{b}$, where a and b are positive integers and b is not divisible by the square of any prime. Find $a+b$.

Reasoned about geometry and inequalities for 2m 47s ›

We will show that the (unique) finite region is a triangle. In fact, one may prove that among the four half-planes into which the two equalities

$$x-yz=y-zx \quad \text{and} \quad y-zx=z-xy,$$

split the plane $x+y+z=75$, exactly one of the four resulting sets is bounded. (In fact, a short analysis shows that writing

$$x-yz=y-zx \implies (y-x)(z-1)=0,$$

and

$$y-zx=z-xy \implies (z-y)(1+x)=0,$$

one deduces that the boundary lines are

$$L_1:\ z=1, \quad L_2:\ x=-1, \quad L_3:\ y=x, \quad L_4:\ y=z.$$

A careful discussion of the four cases – according to whether $z-1$ and $1+x$ are positive or negative – shows that the unique bounded (finite) region is given by

$$(y-x)(z-1)>0 \quad \text{and} \quad (z-y)(1+x)>0 \quad \text{with} \quad x+y+z=75,$$

which (when one accounts for the signs) is equivalent to the inequalities

$$1+x>0, \quad z-1>0, \quad y-x>0, \quad z-y>0,$$

i.e.

$$x>-1, \quad z>1, \quad x<y<z.$$

When one then "peels off" the boundary by considering the two equalities (which occur when either $y=x$ or $z=y$), one finds that the boundary lines of the finite region are exactly

$$x=-1, \quad y=x, \quad y=z.$$

A short computation shows that on the plane $x+y+z=75$ the three lines $x=-1$, $y=x$, and $y=z$ meet at the points

$$A=(-1,-1,77), \quad B=(25,25,25), \quad C=(-1,38,38).$$

These three points form a triangle that is the finite region in question.

To compute its area we may use the cross product. Define

$$\vec{AB}=B-A=(25-(-1),\ 25-(-1),\ 25-77)=(26,26,-52),$$

$$\vec{AC}=C-A=(-1-(-1),\ 38-(-1),\ 38-77)=(0,39,-39).$$

Their cross product is

$$\vec{AB}\times\vec{AC}=(1014,\ 1014,\ 1014),$$

so that

$$\|\vec{AB}\times\vec{AC}\|=1014\sqrt{3}.$$

Thus the area of the triangle is

$$\text{Area}=\frac{1}{2}\|\vec{AB}\times\vec{AC}\|=\frac{1014\sqrt{3}}{2}=507\sqrt{3}.$$

Since the area is given in the form $a\sqrt{b}$ with $a=507$ and $b=3$, the final answer is

$$a+b=507+3=510.$$

Thus, the answer is $\boxed{510}$.

Verifiers

- Note that verifiers don't need to just be answer checks
  - For example, we can write unit tests for code and use them for verification
  - Plus, lots of these out there!
  - As a result, much of **RLVR** is aimed at math and code

```
[TestMethod]
public void TestZoom()
{
  bool gestureDetected = false;
  var threadHolder = new AutoResetEvent(false);

  GestureTestFramework.Validate("Zoom", "TouchInteraction02",
    // On successful gesture detection
    (sender, e) =>
    {
      gestureDetected = true;
      if (e.Error == null)
      {
        var distanceChanged = e.Values.Get<DistanceChanged>();
        // User defined validation code
      }
      else
      {
        Assert.Fail(e.Error.Message);
      }
    });
```

Verifiers: Challenges

- There is still a good amount of complexity…
  - Even if we have a ground truth answer we can check,
  - Might have **multiple forms**
  - **Example:** mathematical expressions

$$\omega = \frac{1}{\frac{\hbar}{E} + \frac{\hbar}{mc^2}(1 - \cos\theta)}$$

  - Check character-by-character? **Too strict**: 1/x and $\frac{1}{x}$ are the same!

Verifiers: Forms

- Ground truth answers with **multiple forms**
  - **Example:** mathematical expressions

- Solution: Implement as functions (programs)
  - Check a set of values

Model Answer:

$$\omega = \frac{1}{\frac{\hbar}{E} + \frac{\hbar}{mc^2}(1 - \cos\theta)}$$

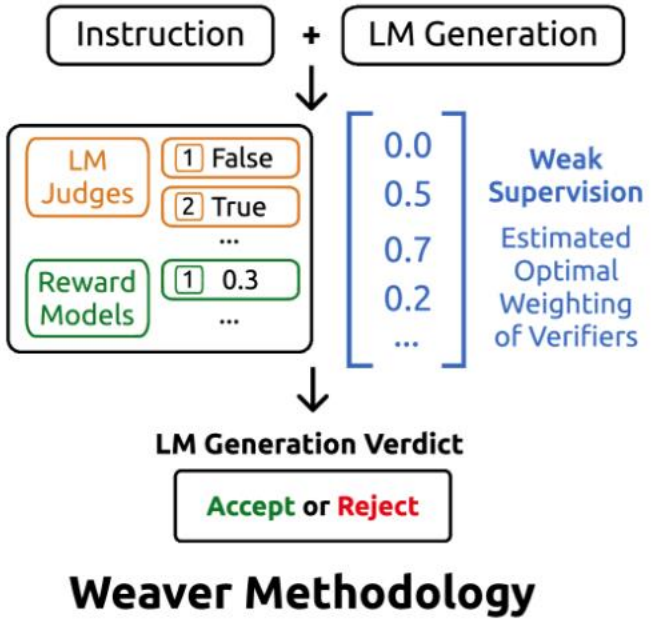
```
import math
def omega_scattered(E: float, m_e:float, theta:float, c:float, h_bar:float) -> float:
    return 1/(h_bar/E + h_bar/(m_e*c**2)*(1-math.cos(theta)))
```

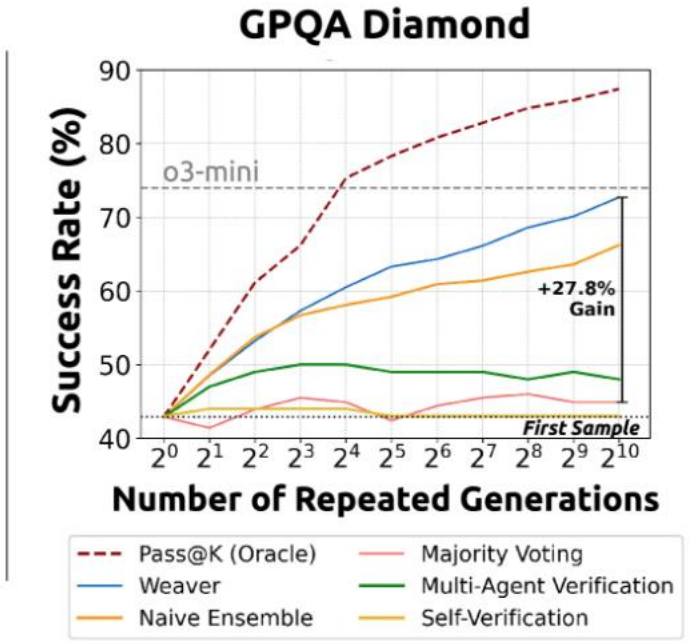
Verifiers: Alternatives

- Sometimes we don't have a straightforward ground-truth check
  - **We can use alternatives!**

Example: "**weak**" **verifiers**

- Rules, partial verifiers, programs, etc.
- Can combine these with "aggregation" rules, just like we did for chain-of-thought





Interlude: Verifiers for Test-Time Scaling

- So far we've been talking about verifiers for **training** a model
  - **I.e., use in RL-based training**

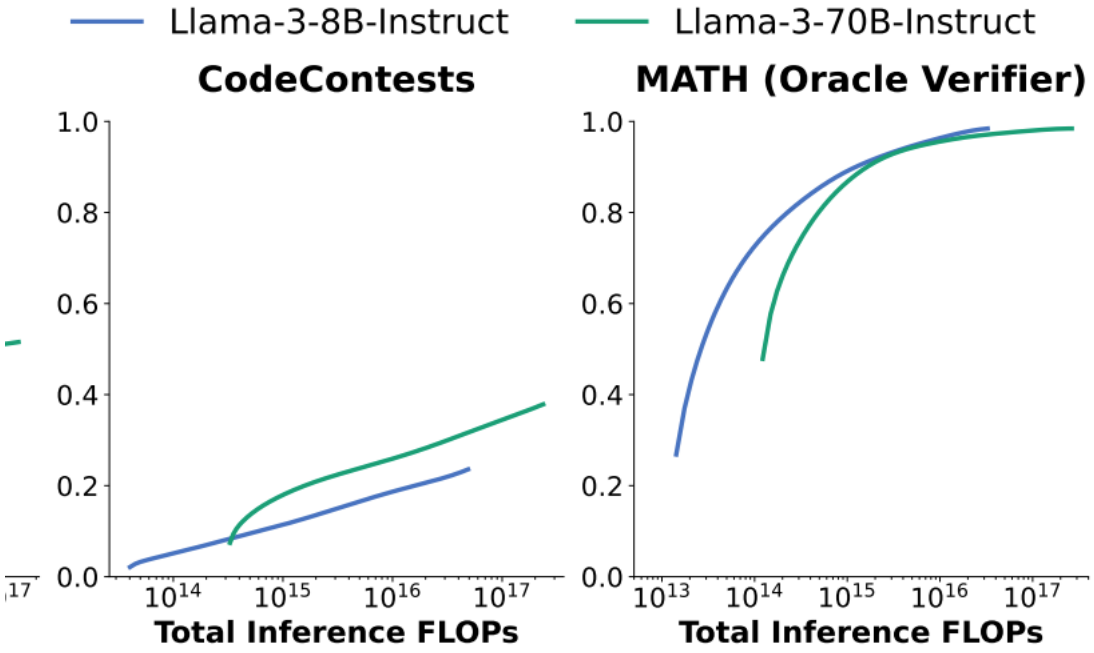
- We can also use them at **inference time**. Keep asking a model the same question & verify response until correct.
  - If the verifier is perfect → first correct response hit
  - If imperfect, we'll lose a bit of overall performance

Why? Interesting tradeoff here:

- Put compute into training a **bigger** model and doing **fewer attempts** per question **OR** train a **smaller model** and put more compute into **repeats**?

Interlude: Verifiers for Test-Time Scaling

- Put compute into training a **bigger** model and doing **fewer attempts** per question **OR** train a **smaller model** and put more compute into **repeats**?
  - Slightly more complex too: each question + answer is more expensive on a bigger model
  - No single right answer: optimum is task dependent!



Brown et al ‘24

Back to Verifier Alternatives

- Sometimes we don't have a straightforward ground-truth check
  - **We can use alternatives!**

- In addition to functions/programs/lookup tables, we can also use **models** (neural networks) as verifiers
- Two broad classes of choices,
  - **"Reward" models**
    - Note: slghtly different from the RLHF reward models
    - Train a model that outputs scores that **indicates correctness**
  - **LLM-as-a-judge**
    - Use an LLM and ask it if a response is correct

Back to Verifier Alternatives

In addition to functions/programs/lookup tables, we can also use **models** (neural networks) as verifiers

- **"Reward" models**
- Further subdivided into two types,
- 1. "**Outcome**" reward models (ORMs)
  - Directly predict correctness of final outcome
- 2. "Process" reward models (PRMs)
  - Try to predict correctness of intermediate steps

- PRMs have more promise (assign credit to individual solution) but are harder to build

Break & Questions

Outline

- **Verifiers**
  - Notion of “verifiers”, challenges for oracle verifiers, alternatives & weak verifiers, test-time scaling, ORMs and PRMs
- **RL with Verifiers**
  - Objectives, more on PPO, group-relative policy optimization (GRPO), reward construction, applications

Back to RL: RLHF → RLVR

When we did RLHF, we used a particular approach to RL

- Our goal: make our model (the "policy" in an RL sense) more likely to produce high-reward output
- We will now swap the reward
  - **Before**: preference model for alignment with humans
  - **Now**: verifiable rewards for reasoning
    - Using the verifiers that we have built
- Our starting point: use PPO (the method for RLHF) and make some adjustments

PPO Details

- We used PPO to do RLHF
- Can still do this and integrate a notion of verifier correctness into the reward
- Let's dive a bit deeper into **PPO:** another formulation:

$$\hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$

- Two forms

(that we can combine)

Advantage $A_t = Q(s_t, a_t) - V(s_t)$

$$\hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

PPO Details: Advantages

- Let's dive a bit deeper into **PPO:** another formulation:

$$\hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$

Advantage $A_t = Q(s_t, a_t) - V(s_t)$

What are these **Qs** and **Vs?**

- Standard objects in RL: Q function and V (value function)
- Roughly, $\boldsymbol{Q(s_t, a_t)}$ is the exp. cumulative value from state st by taking an action at, while $\boldsymbol{V(s_t)}$ is exp. cumulative value by following policy (rather than at) specifically
- So we're computing **benefit of producing token** a_t

PPO to GRPO

- An improved version: **GRPO** (**G**roup **R**elative **P**olicy **O**ptimization)
  - Shao et al, DeepSeekMath

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right).$$

- Most elements are the same compared to PPO, but note that we sample a **group** of G responses.
- Advantage:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}$$

PPO to GRPO

Most elements are the same compared to PPO, but note that we sample a **group** of G responses.

- Advantage:

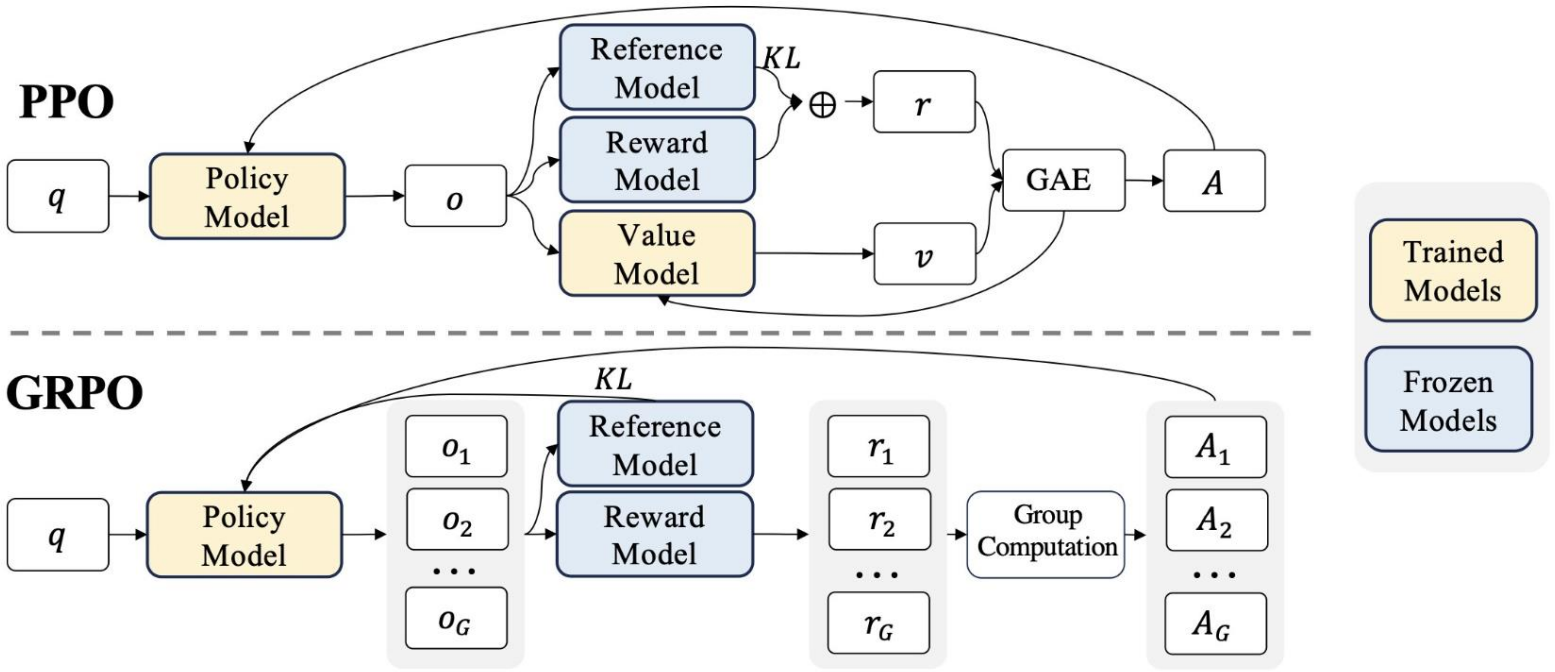
$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}$$

- Note: very simple compared to having to work with value functions. Our advantage just comes down to **getting problems right more often** (relative to group of responses)
  - Still need to define r, the reward

PPO to GRPO: Diagram

Note: very simple compared to having to work with value functions. Our advantage just comes down to **getting problems right more often** (relative to group of responses)

- Still need to define r, the reward



Sergio Paniego

GRPO/DeepSeek R1 Rewards

- How to use verifiers in rewards?

$$A_i = \frac{r_i - \mathrm{mean}(\{r_1, r_2, \cdots, r_G\})}{\mathrm{std}(\{r_1, r_2, \cdots, r_G\})}$$

Very simple: DeepSeek R1 uses:

- **Accuracy rewards**: The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards**: In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between '<think>' and '</think>' tags.

Note the format reward---encourages reasoning!

GRPO Example

Let's work out an example

- Advantage:

$$A_i = \frac{r_i - \mathrm{mean}(\{r_1, r_2, \cdots, r_G\})}{\mathrm{std}(\{r_1, r_2, \cdots, r_G\})}$$

Example: prompt: *Solve for* x: $2x + 3 = 11$

Run prompt 4 times, get:

1. 2x + 3 = 11 ⇒ 2 x = 8 ⇒ x = 4
2. 2 x + 3 = 11 ⇒ 2 x = 14 ⇒ x = 7
3. Subtract 3: 2 x = 8 2x=8. Divide by 2: x = 4
4. x = 5

Rewards: [1,0,1,0], **Relative advantages**: A = [0.5, −0.5, +0.5, −0.5]

RLVR Produces Strong Performance on Math

AIME: very popular math competition

Overall | **AIME 2025 I** | AIME 2025 II | HMMT February 2025 | USAMO 2025

Model	Acc	Cost	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
gemini-2.5-pro ⚠	83.33%	N/A															
o3-mini (high)	80.00%	$3.19															
o1 (medium)	78.33%	$44.40															
o3-mini (medium)	73.33%	$1.67															
DeepSeek-R1	65.00%	$4.91															
QwQ-32B ⚠	60.00%	$1.24															
DeepSeek-V3-03-24 ⚠	53.33%	$0.25															
o3-mini (low)	53.33%	$0.62															
DeepSeek-R1-Distill-32B	53.33%	N/A															
gemini-2.0-flash-thinking	51.67%	N/A															
DeepSeek-R1-Distill-14B	50.00%	$1.15															
DeepSeek-R1-Distill-70B	50.00%	$1.35															
Claude-3.7-Sonnet (Think) ⚠	46.67%	$22.17															
QwQ-32B-Preview	36.67%	$0.58															
gemini-2.0-flash	30.00%	$0.06															

https://matharena.ai/

And Physics

Theoretical Physics Benchmark (TPBench) - a Dataset and Study of AI Reasoning Capabilities in Theoretical Physics

Daniel J.H. Chung[1], Zhiqi Gao[2], Yurii Kvasiuk[1], Tianyi Li[1], Moritz Münchmeyer[1,5], Maja Rudolph[3], Frederic Sala[2], and Sai Chaitanya Tadepalli[4]

[1]Department of Physics, University of Wisconsin-Madison
[2]Department of Computer Science, University of Wisconsin-Madison
[3]Data Science Institute (DSI), University of Wisconsin-Madison
[4]Department of Physics, Indiana University, Bloomington
[5]NSF-Simons AI Institute for the Sky (SkAI), Chicago

February 25, 2025

Abstract

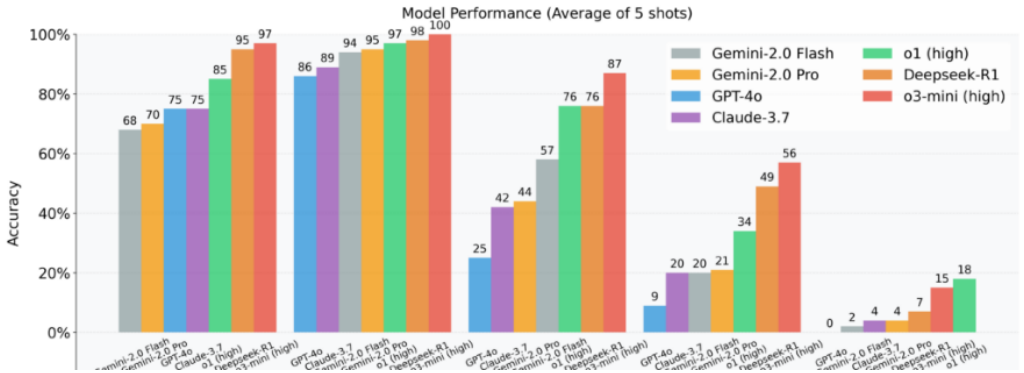
We introduce a benchmark to evaluate the capability of AI to solve problems in theoretical physics, focusing on high-energy theory and cosmology. The first iteration of our benchmark consists of 57 problems of varying difficulty, from undergraduate to research level. These problems are novel in the sense that they do not come from public problem collections. We evaluate our data set on various open and closed language models, including o3-mini, o1, DeepSeek-R1, GPT-4o and versions of Llama and Qwen. While we find impressive progress in model performance with the most recent models, our research-level difficulty problems are mostly unsolved. We address challenges of auto-verifiability and grading, and discuss common failure modes. While currently state-of-the art models are still of limited use for researchers, our results show that AI assisted theoretical physics research may become possible in the near future. We discuss the main obstacles towards this goal and possible strategies to overcome them. The public problems and solutions, results for various models, and updates to the data set and score distribution, are available on the website of the dataset tpbench.org.

TP Bench – Theoretical Physics Benchmark for AI

TPBench is a curated dataset and evaluation suite designed to measure the reasoning capabilities of AI models in theoretical physics. Our test problems span multiple difficulty levels—from undergraduate to frontier research—and cover topics such as cosmology, high-energy theory, general relativity, and more. By providing a unified framework for problem-solving and auto-verifiable answers, TPBench aims to drive progress in AI-based research assistance for theoretical physics.

Read the TPBench Paper on arxiv
Access Public Dataset on Huggingface

Current Model Performance



Thank You!