# Large reasoning models & Test-time Scaling

## CS 4804: Introduction to AI

*Fall 2025*

https://tuvllms.github.io/ai-fall-2025/

**Tu Vu**

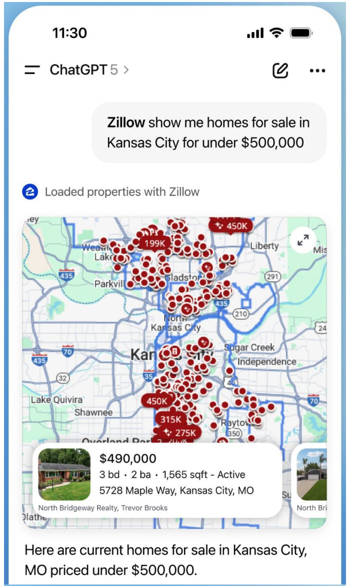
VIRGINIA TECH

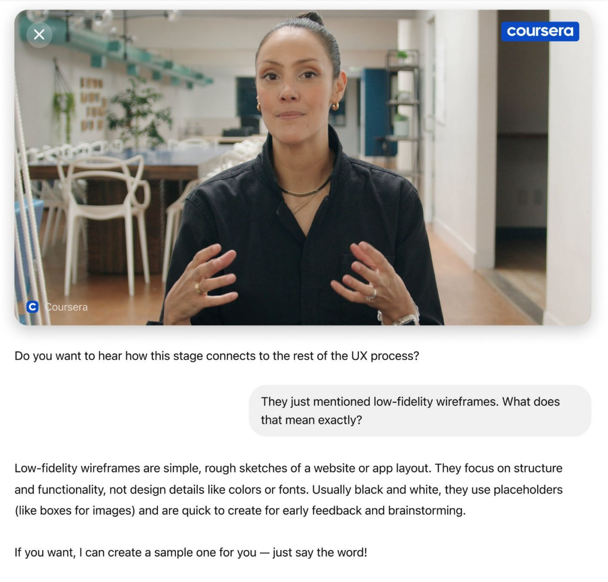
# Logistics

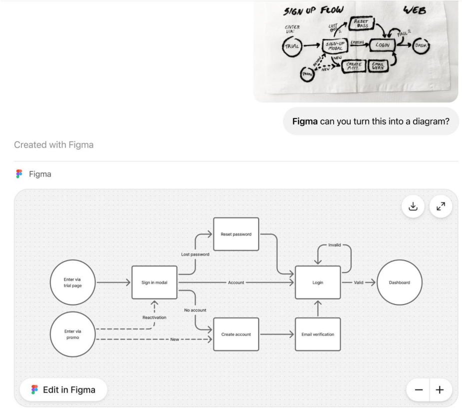
- Homework 1 due today
- Final Project
  - Curating data (optional)
  - Benchmarking models
  - 0-shot/few-shot/chain-of-thought/self-consistency prompting
  - Test-time scaling
- Final presentations: 12/4 & 12/9

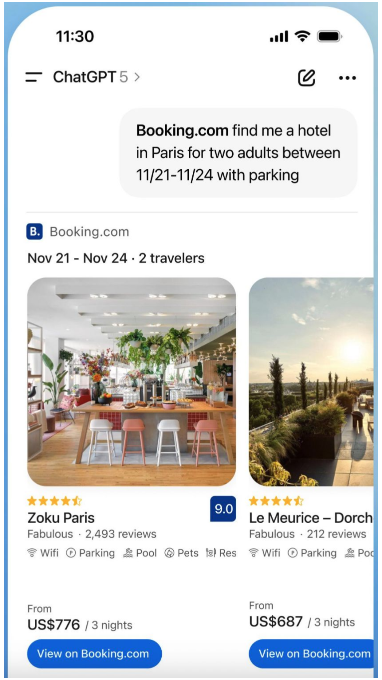
# OpenAI DevDay 2025

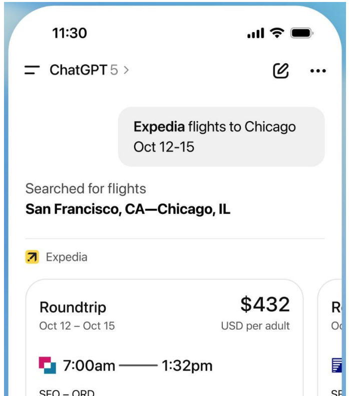
- Apps in ChatGPT

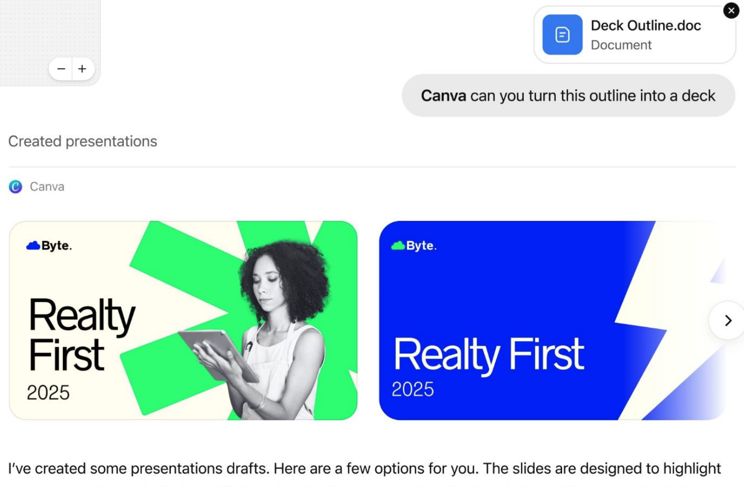


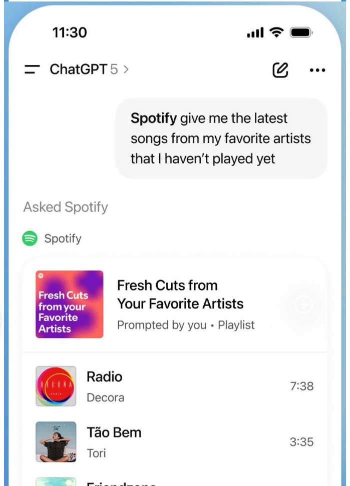



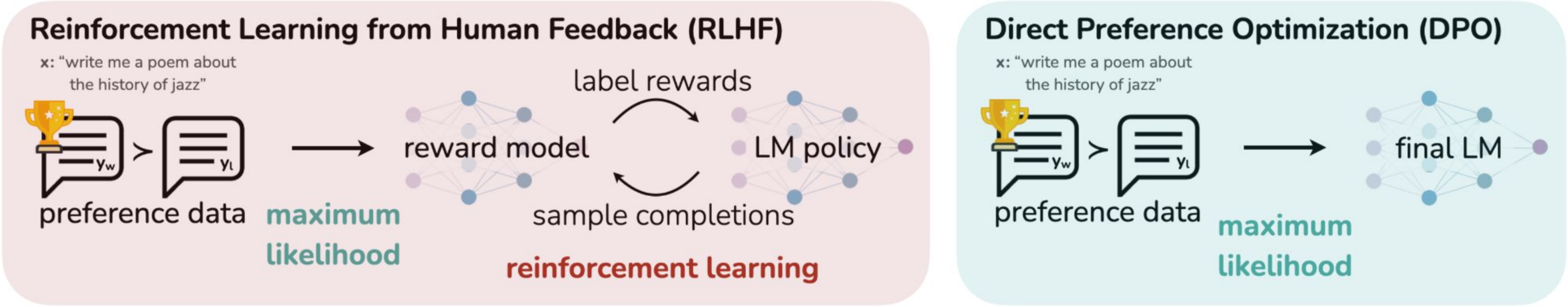



# Video generation with Sora 2

- https://x.com/OpenAI/status/1973143639200243959

# RLHF vs. DPO



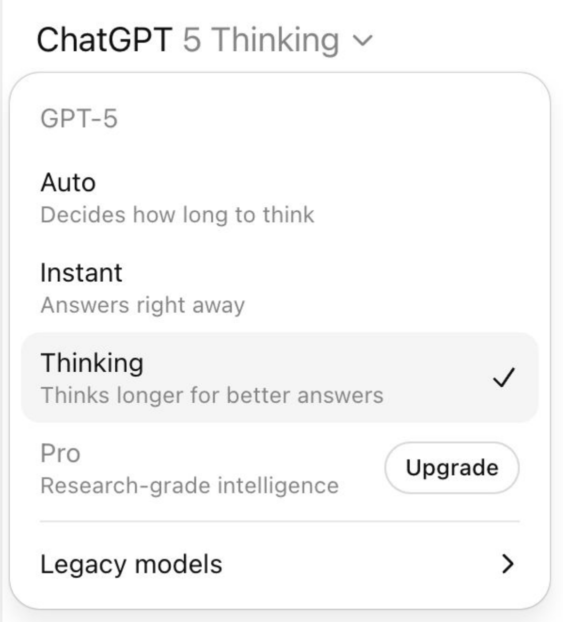

**RLHF**

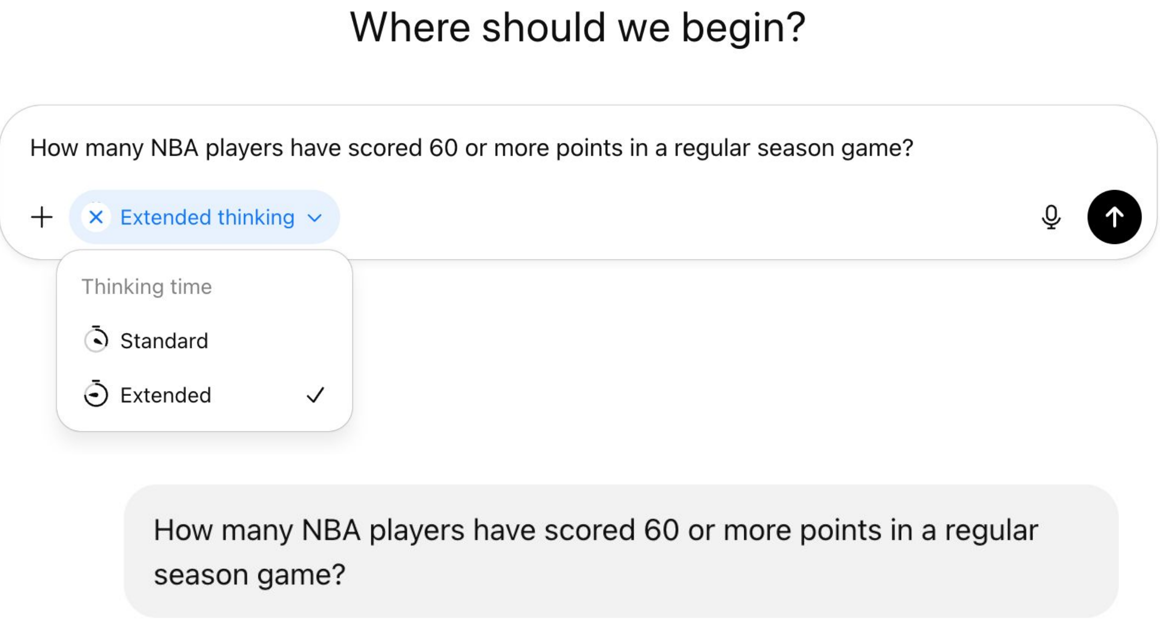
$$\max_{\pi} \mathbb{E}_{x\sim D, y\sim\pi(y|x)} \left[r(x,y)\right] - \beta D_{\mathrm{KL}}\left[\pi(y|x) \,||\, \pi_{\mathrm{ref}}(y|x)\right]$$

**DPO**

$$L_{\mathrm{DPO}}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\mathrm{ref}}(y_l \mid x)}\right)\right]$$

# Large reasoning models (LRMs) / Thinking models



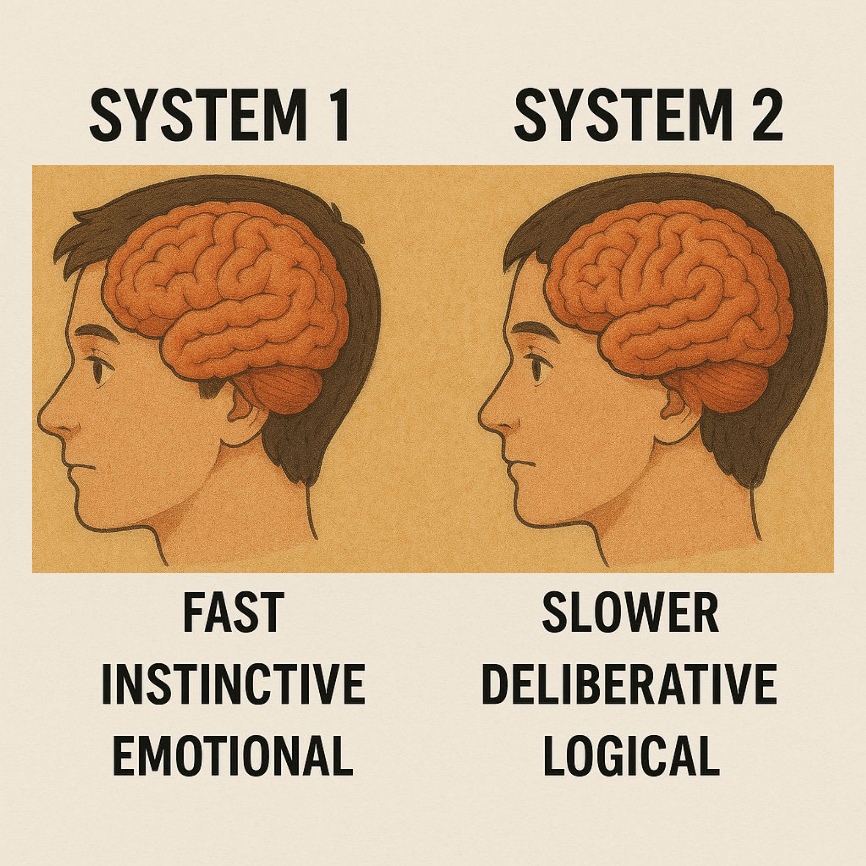

# THINKING, FAST AND SLOW

DANIEL KAHNEMAN

WINNER OF THE NOBEL PRIZE IN ECONOMICS

# System 1 & System 2

**Which is better?**




ChatGPT image

# OpenAI's o1

## How it works

We trained these models to spend more time thinking through problems before they respond, much like a person would. Through training, they learn to refine their thinking process, try different strategies, and recognize their mistakes.

Similar to how a human may think for a long time before responding to a difficult question, o1 uses a chain of thought when attempting to solve a problem. Through reinforcement learning, o1 learns to hone its chain of thought and refine the strategies it uses. It learns to recognize and correct its mistakes. It learns to break down tricky steps into simpler ones. It learns to try a different approach when the current one isn't working. This process dramatically improves the model's ability to reason.

# Can we infer its structure and what kinds of behaviors it rewards?

- The reward model likely provides scalar feedback that evaluates both **final answer** (correctness, usefulness, safety, clarity) and **reasoning process** (depth, coherence, reliability, safety awareness)
- Human annotators or automated evaluators review model outputs on reasoning tasks

# What the reward model likely gives high scores for

### 1. Correct reasoning and factual accuracy

Outputs that reach correct conclusions through logically consistent reasoning receive higher rewards.

- On math or logic tasks: correct answers with verifiable reasoning.
- On open-ended questions: responses that are factually accurate and well supported.

### 2. Coherent, interpretable reasoning chains

Even if the chain of thought is not visible, internal reasoning steps that are internally consistent and lead to stable answers are likely rewarded.

- The model learns to reason step by step rather than jump to conclusions.
- RL training can penalize incoherent or contradictory internal reasoning trajectories.

### 3. Efficiency and confidence calibration

The model may get higher reward when it uses an appropriate amount of “thinking” (not too short or too long) and when it expresses uncertainty realistically.

- Correct but overly long reasoning could be slightly penalized.
- Overconfident incorrect answers could get low reward.

# What the reward model likely gives high scores for (cont'd)

### 4. Safety, compliance, and harmlessness

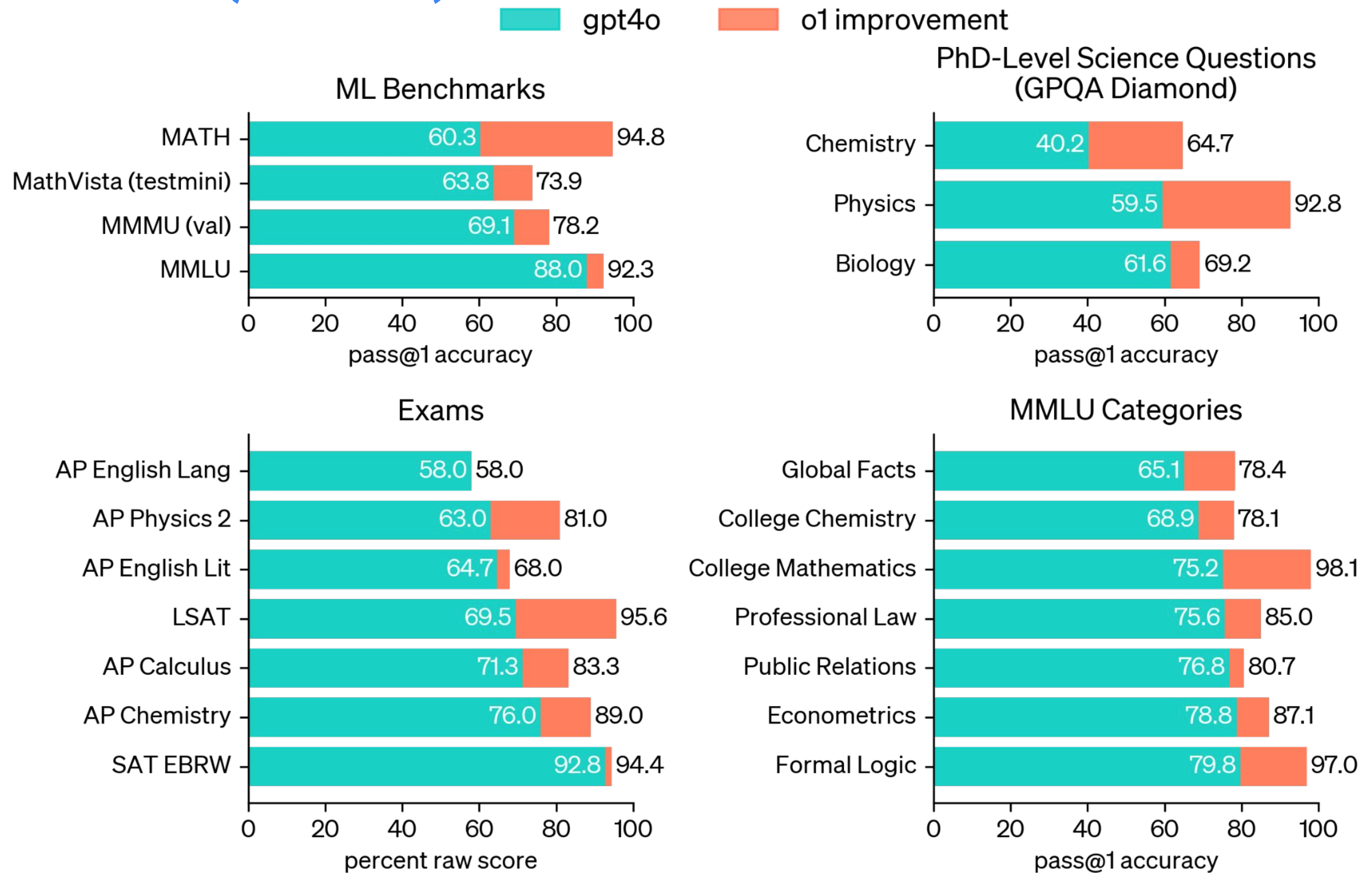
Outputs that adhere to safety policies and avoid unsafe or biased reasoning receive higher reward.

- The o1 system card emphasizes "deliberative alignment," meaning the model is rewarded for reasoning *about* safety before answering.
- Unsafe or policy-violating reasoning steps are penalized.

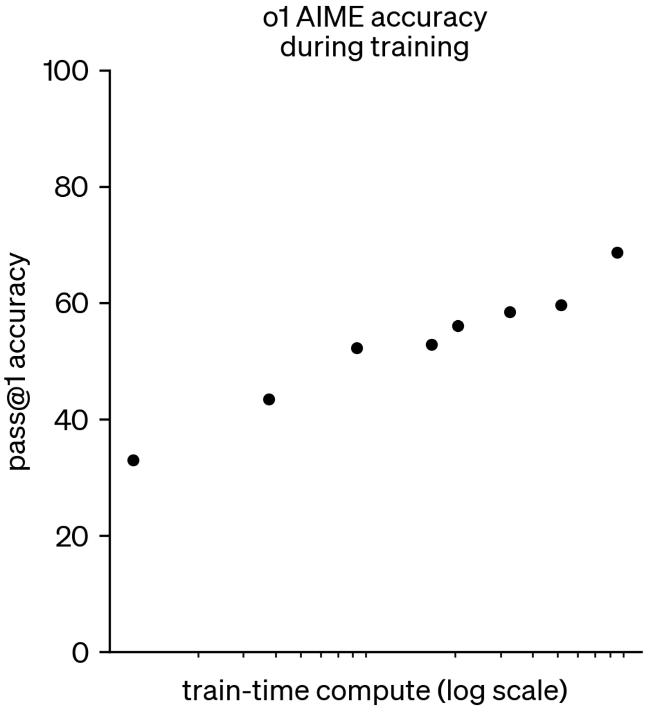
### 5. General helpfulness and clarity

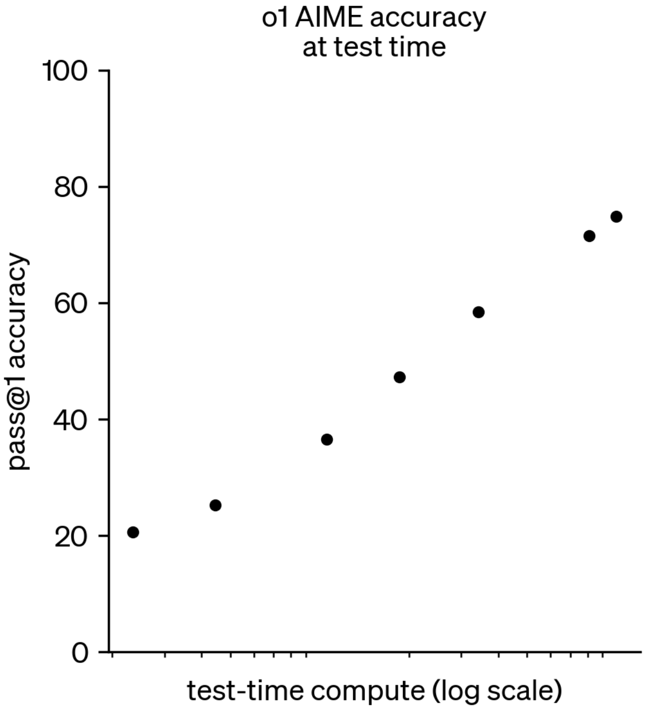
The reward model also encourages clarity, helpful tone, and clear communication of reasoning results, similar to prior RLHF models.

# OpenAI's o1(cont'd)



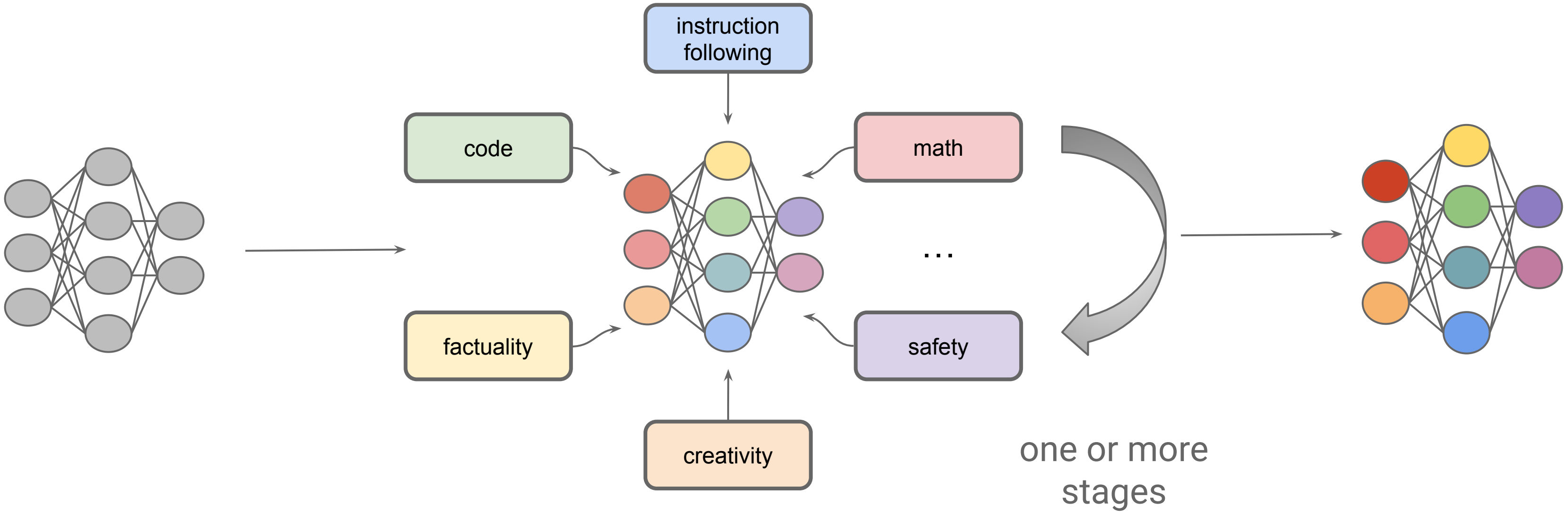
# o1 performance smoothly improves with both train-time and test-time compute




https://openai.com/index/learning-to-reason-with-llms/

# From System 1 to System 2




System 1

Supervised Fine-tuning and/or Reinforcement Learning on long Chain-of-Thought data

System 2

deepseek

# DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning


DeepSeek-AI

research@deepseek.com


## Abstract


We introduce our first-generation reasoning models, DeepSeek-R1-Zero and DeepSeek-R1. DeepSeek-R1-Zero, a model trained via large-scale reinforcement learning (RL) without supervised fine-tuning (SFT) as a preliminary step, demonstrates remarkable reasoning capabilities. Through RL, DeepSeek-R1-Zero naturally emerges with numerous powerful and intriguing reasoning behaviors. However, it encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates multi-stage training and cold-start data before RL. DeepSeek-R1 achieves performance comparable to OpenAI-o1-1217 on reasoning tasks. To support the research community, we open-source DeepSeek-R1-Zero, DeepSeek-R1, and six dense models (1.5B, 7B, 8B, 14B, 32B, 70B) distilled from DeepSeek-R1 based on Qwen and Llama.

# DeepSeek-R1's motivation

- develop reasoning capabilities **without any supervised data (labeled input-output pairs)**
- focus on their **self-evolution** through **a pure reinforcement learning process**

# Reinforcement Learning Algorithm

- **Group Relative Policy Optimization** (GRPO)
  - A variant of RLHF
  - Sample outputs from the current (old) policy and then optimize the policy model by maximizing the scalar reward given by a reward model

# Reinforcement Learning from Verifiable Rewards (RLVR)

The reward is the source of the training signal, which decides the optimization direction of RL. To train DeepSeek-R1-Zero, we adopt a rule-based reward system that mainly consists of two types of rewards:

- **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.

# Guided Chain-of-Thought (CoT) template

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: prompt. Assistant:

# Group Relative Policy Optimization (GRPO)

**Group Relative Policy Optimization** In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question $q$, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model $\pi_\theta$ by maximizing the following objective:

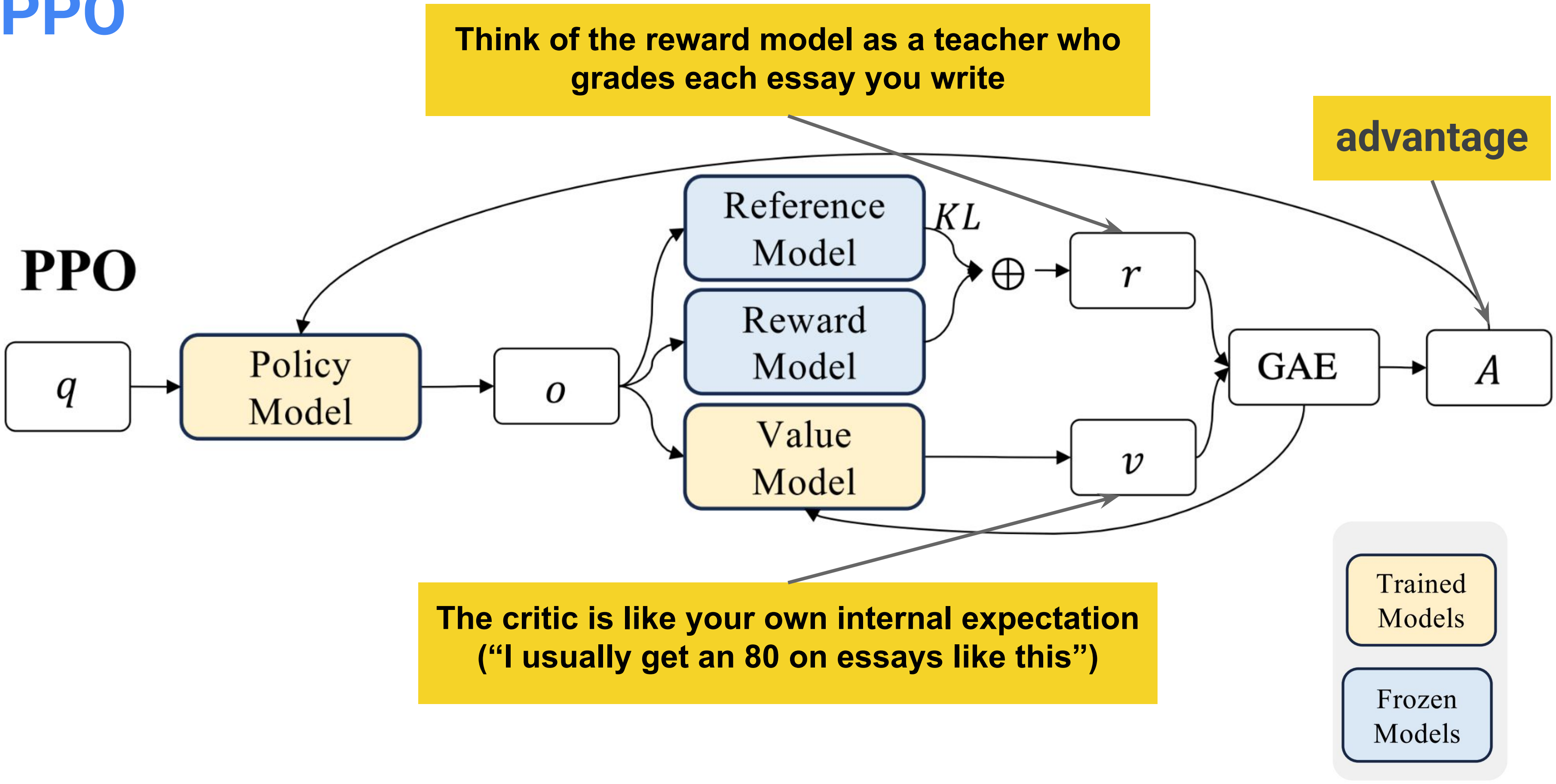
$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right)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

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \tag{2}$$

where $\varepsilon$ and $\beta$ are hyper-parameters, and $A_i$ is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - mean(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$

# PPO


Think of the reward model as a teacher who grades each essay you write
advantage
PPO
q
Policy Model
o
Reference Model
Reward Model
Value Model
KL
r
v
GAE
A
Trained Models
Frozen Models
The critic is like your own internal expectation ("I usually get an 80 on essays like this")

# Group Relative Policy Optimization (GRPO) (cont'd)

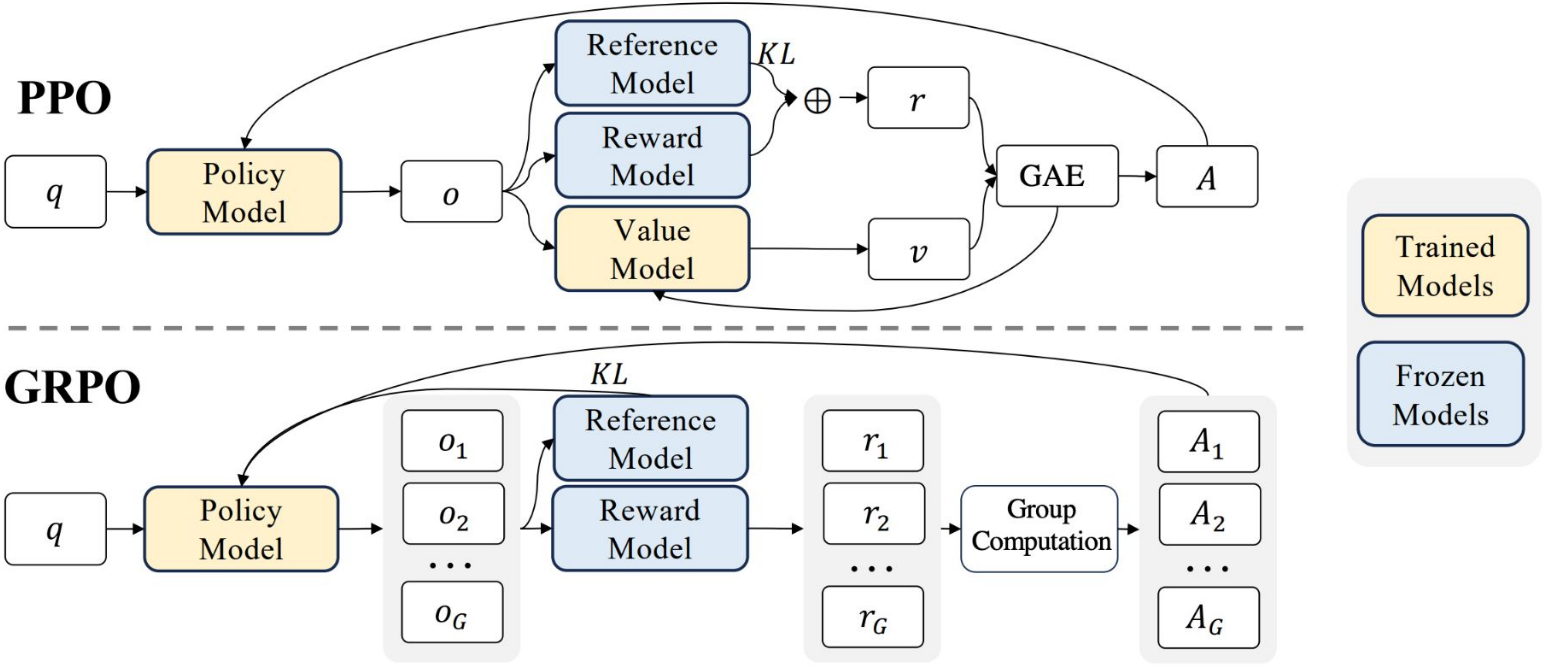



Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

# Group advantage estimation

The relative advantage for each output $o_i$ is defined as:

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

This means that an output is considered “good” if its reward is higher than the group average.

This relative normalization stabilizes updates and avoids the need for an explicit baseline network.

# Maximizing expected reward

The primary objective is to make outputs with higher rewards (from a reward model or human feedback) more likely under the updated policy.

This is achieved through the ratio term

$$\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i,$$

which increases the probability of samples with positive advantages $A_i > 0$ and decreases it for those with negative advantages $A_i < 0$.

Thus, GRPO pushes the model to favor responses that receive relatively higher scores within each group.

# Maintaining policy stability

The **clipping function**

$$\operatorname{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)$$

prevents large updates that could destabilize the model.

This is inherited from PPO: it ensures that the new policy does not deviate too aggressively from the previous one in a single training step.

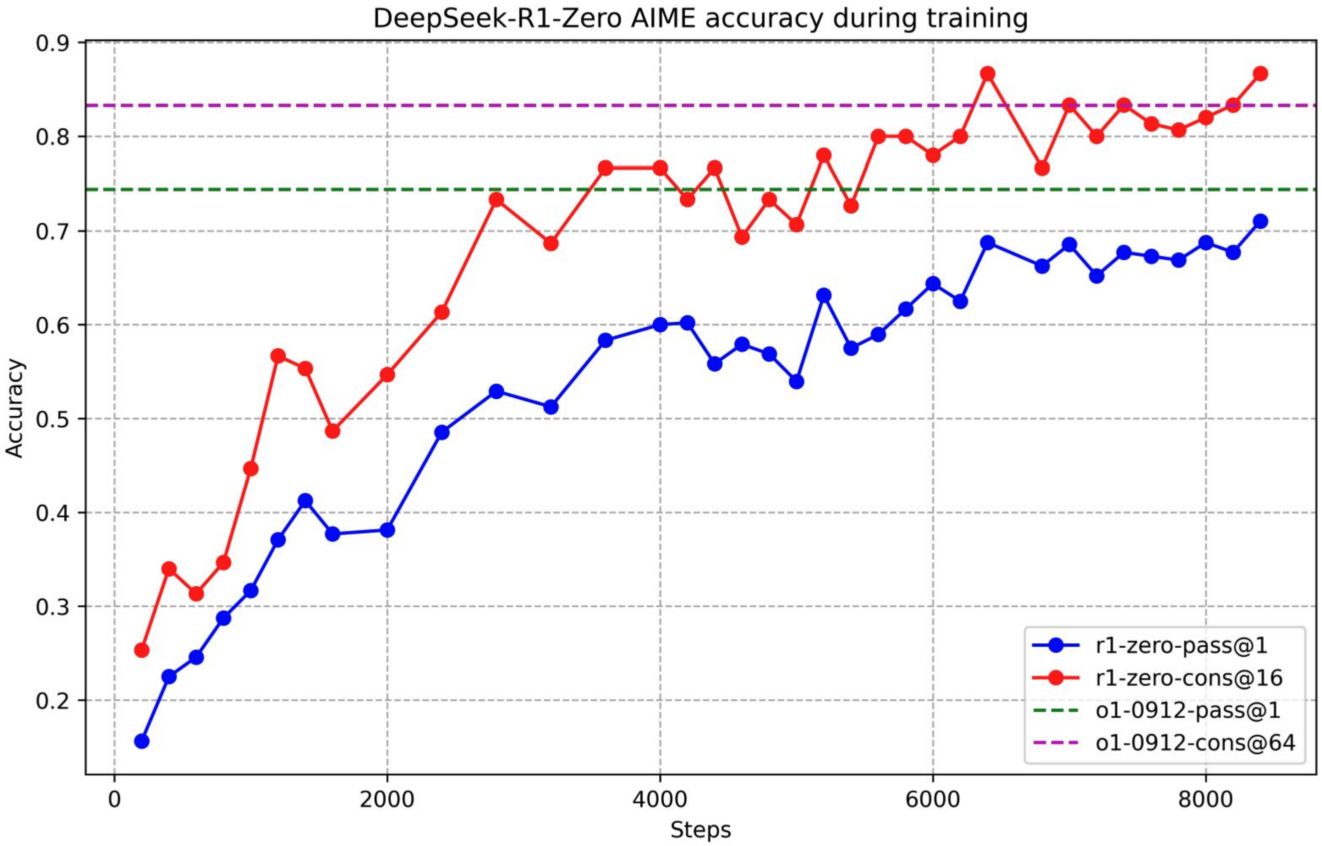
# Controlling divergence from the reference model

The **KL penalty term**

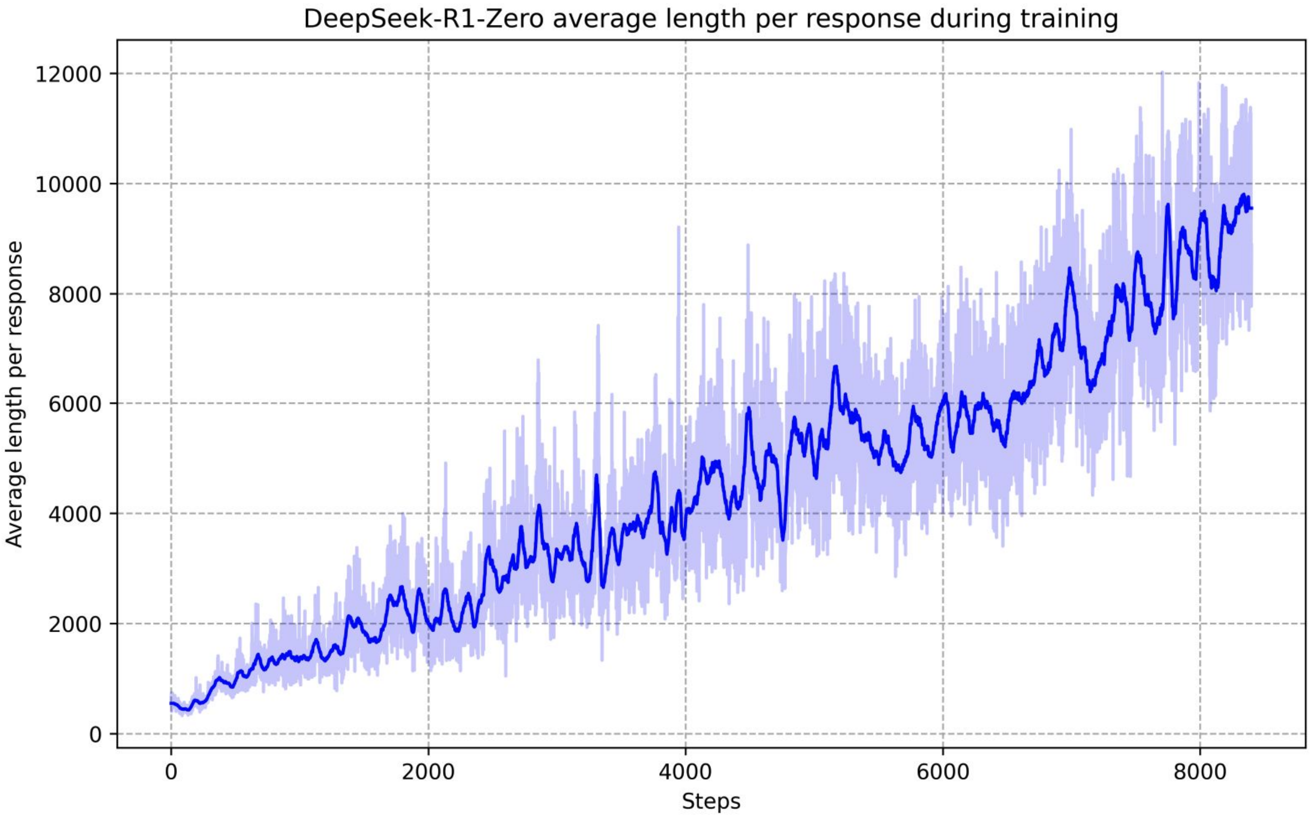
$$\beta D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$$

discourages the policy from diverging too far from a stable reference (often the supervised fine-tuned model).
This constraint keeps the learned behavior aligned with the base model's general language ability and prevents reward hacking or degeneration.

# Accuracy of DeepSeek-R1-Zero during training



# DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time



# An interesting "aha moment" of DeepSeek-R1-Zero

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both $\cdots$

$\left(\sqrt{a - \sqrt{a + x}}\right)^2 = x^2 \implies a - \sqrt{a + x} = x^2.$

Rearrange to isolate the inner square root term:

$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$

...

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$

We started with the equation:

$\sqrt{a - \sqrt{a + x}} = x$

First, let's square both sides:

$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$

Next, I could square both sides again, treating the equation: $\cdots$

...

# Problems with DeepSeek-R1-Zero

- It encounters challenges such as poor readability, and language mixing

# Problems with DeepSeek-R1-Zero

- It encounters challenges such as poor readability, and language mixing

# DeepSeek-R1

- Includes a small amount of cold-start data (thousands of long Chain-of-Thought (CoT) examples)

Post

**lmarena.ai (formerly lmsys.org)**
@lmarena_ai

Breaking News: DeepSeek-R1 surges to the top-3 in Arena🐳!

Now ranked #3 Overall, matching the top reasoning model, o1, while being 20x cheaper and open-weight!

Highlights:
- #1 in technical domains: Hard Prompts, Coding, Math
- Joint #1 under Style Control
- MIT-licensed

A massive congrats to @deepseek_ai for this incredible milestone and gift to the community! More analysis below 👇

Category: Overall

Apply filter: Style Control | Show Deprecated

Overall Questions
#models: 195 (100%) #votes: 2,572,591 (100%)

**DeepSeek-R1 #3 in Arena Overall!**

| Rank* (UB) | Rank (StyleCtrl) | Model | Arena Score | 95% CI | Votes | Organization |
|---|---|---|---|---|---|---|
| 1 | 1 | Gemini-Exp-1206 | 1374 | +5/-4 | 22116 | Google |
| 1 | 3 | Gemini-2.0-Flash-Thinking-Exp-01-21 | 1382 | +8/-6 | 6437 | Google |
| 3 | 1 | ChatGPT-4o-latest (2024-11-20) | 1365 | +4/-4 | 35328 | OpenAI |
| 3 | 1 | DeepSeek-R1 | 1357 | +12/-13 | 1883 | DeepSeek |
| 4 | 1 | o1-2024-12-17 | 1352 | +6/-6 | 9230 | OpenAI |
| 4 | 5 | Gemini-2.0-Flash-Exp | 1356 | +4/-4 | 20939 | Google |

# Using reasoning models

## Get started with reasoning

Call the Responses API and specify your reasoning model and reasoning effort:

Using a reasoning model in the Responses API

python

```python
from openai import OpenAI

client = OpenAI()

prompt = """
Write a bash script that takes a matrix represented as a string with
format '[1,2],[3,4],[5,6]' and prints the transpose in the same format.
"""

response = client.responses.create(
    model="gpt-5",
    reasoning={"effort": "medium"},
    input=[
        {
            "role": "user",
            "content": prompt
        }
    ]
)

print(response.output_text)
```

**For GPT-oss: the reasoning level can be set in the system prompts, e.g., “Reasoning: high”.**

# Code

unsloth

## ✨ Train for Free

Notebooks are beginner friendly. Read our guide. Add dataset, click "Run All", and export your trained model to GGUF, Ollama, vLLM or Hugging Face.

| Unsloth supports | Free Notebooks | Performance | Memory use |
|---|---|---|---|
| **gpt-oss (20B)** | ▶ Start for free | 1.5x faster | 70% less |
| **Gemma 3n (4B)** | ▶ Start for free | 1.5x faster | 50% less |
| **Qwen3 (14B)** | ▶ Start for free | 2x faster | 70% less |
| **gpt-oss (20B): GRPO** | ▶ Start for free | 2x faster | 80% less |
| **Qwen2.5-VL (7B): GSPO** | ▶ Start for free | 1.5x faster | 80% less |
| **Phi-4 (14B)** | ▶ Start for free | 2x faster | 70% less |
| **Llama 3.2 Vision (11B)** | ▶ Start for free | 2x faster | 50% less |
| **Llama 3.1 (8B)** | ▶ Start for free | 2x faster | 70% less |
| **Mistral v0.3 (7B)** | ▶ Start for free | 2.2x faster | 75% less |
| **Orpheus-TTS (3B)** | ▶ Start for free | 1.5x faster | 50% less |

- See all our notebooks for: Kaggle, GRPO, **TTS** & Vision
- See all our models and all our notebooks
- See detailed documentation for Unsloth here

# Thank you!