W2.2- Reasoning LLM and GRPO
– Advanced

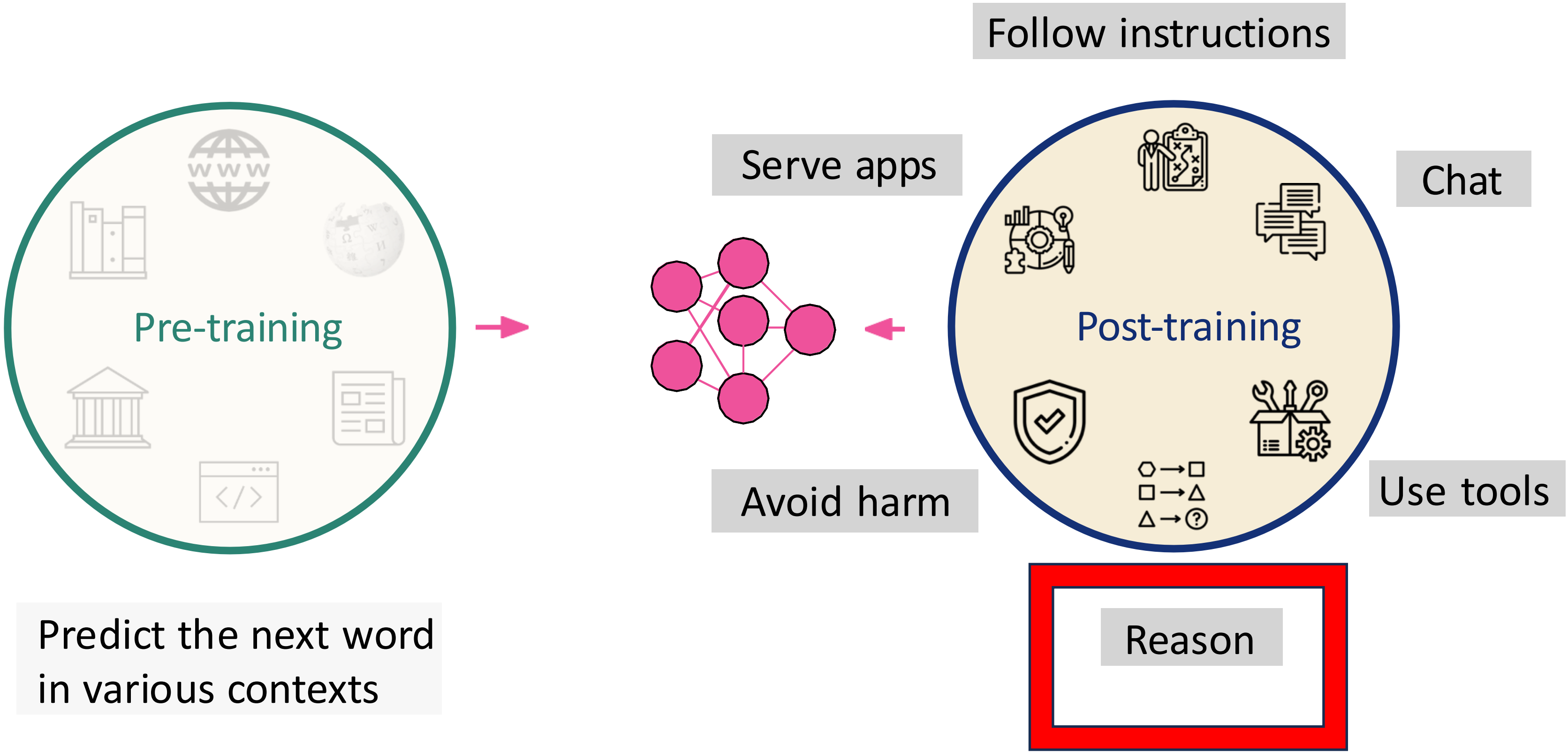
2026 Spring

LLM Agents Foundation & Applications

Dr. Yanjun Qi

20260225

Building a modern LLM



Pre-training

Predict the next word
in various contexts

Post-training

Follow instructions

Serve apps

Chat

Avoid harm

Use tools

Reason

Credit: Ai2

What is LLM reasoning?

- What is LLM reasoning?
  - Multi-step inference beyond pattern completion
  - Constructing intermediate reasoning steps
  - Structured transformation from input to output

- Types of Reasoning in LLMs
  - Logical reasoning
  - Mathematical reasoning
  - Planning / sequential reasoning
  - Causal & commonsense reasoning

Three Levels of LLM Capability

- Level 1: Surface completion
- Level 2: Structured inference
- Level 3: Deliberative planning
  - Modern reasoning models aim for Level 3

LLM reasoning = structured multi-step inference

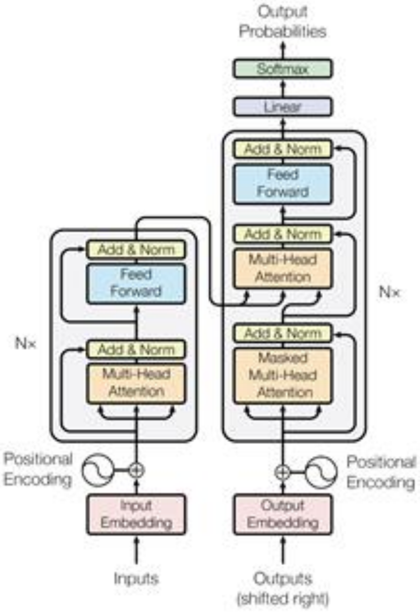
- Necessary for planning, robustness, safety
- Critical for agentic AI systems
- Key frontier in AI research

This Class:

- Review: RLHF and PPO
- Paper1: Reasoning with Deep Seek-R1
- Paper 2: Reasoning Language Models: A Blueprint

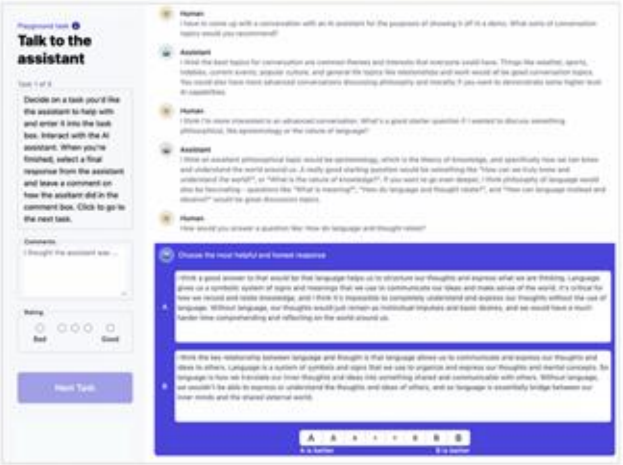
Review: RLHF Basics and PPO

Three phases of RLHF



Vaswani et al. 2017

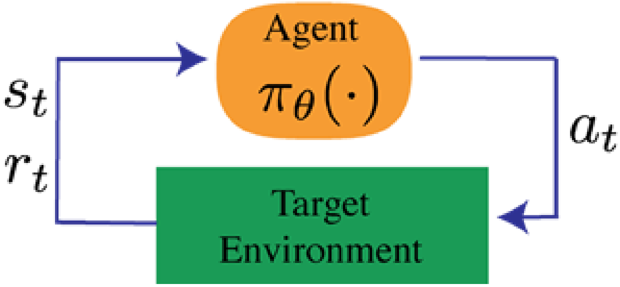
1. base model (instruction, helpful, chatty etc.)



Talk to the assistant

Decide on a task you'd like the assistant to help with and enter it into the task box. Interact with the AI assistant. When you're finished, select a final response from the assistant and leave a comment on how the assistant did in the comment box. Click to go to the next task.

2. preference collection & reward model training



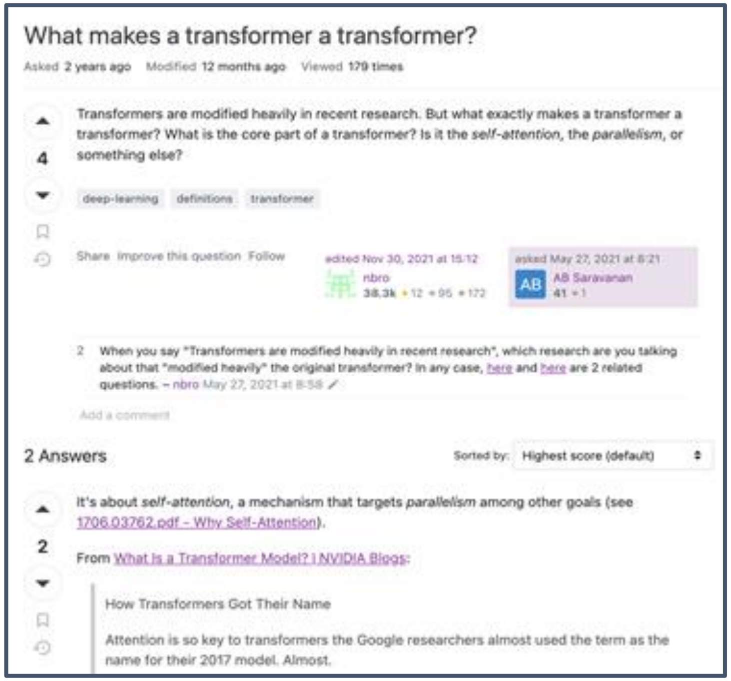
3. reinforcement learning [policy optimization

1. Instruction-tuned language model

starting point: a base language model

continue training a transformer with pairs of

question: answer fine training with (x_t, y_t)



What makes a transformer a transformer?
Asked 2 years ago Modified 12 months ago Viewed 179 times
4
deep-learning
definitions
transformer
Share Improve this question Follow
edited Nov 30, 2021 at 15:12
nbro
38.3k 12 95 172
asked May 27, 2021 at 8:21
AB
AB Saravanan
41 1
Add a comment
2 Answers
Sorted by: Highest score (default)
2

Transformers are modified heavily in recent research. But what exactly makes a transformer a transformer? What is the core part of a transformer? Is it the *self-attention*, the *parallelism*, or something else?

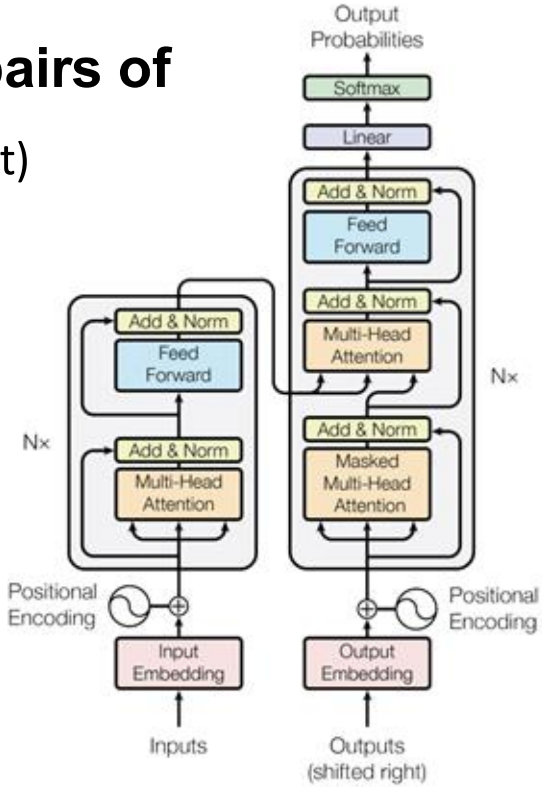
2 When you say "Transformers are modified heavily in recent research", which research are you talking about that "modified heavily" the original transformer? In any case, here and here are 2 related questions. – nbro May 27, 2021 at 8:58

It's about *self-attention*, a mechanism that targets *parallelism* among other goals (see 1706.03762.pdf - Why Self-Attention).

From What Is a Transformer Model? | NVIDIA Blogs:

> How Transformers Got Their Name
>
> Attention is so key to transformers the Google researchers almost used the term as the name for their 2017 model. Almost.

Stack Overflow :*What makes a transformer a transformer?*, nbro 2021

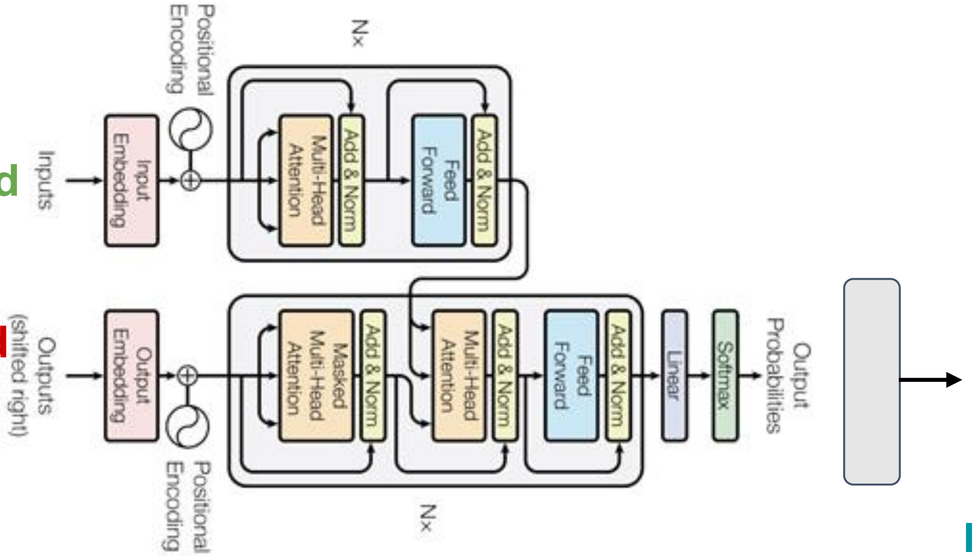


2. Reward /Preference model structure

input pair:

prompt + selected completion

prompt + rejected completion



The Transformer - Vaswani et al. 2017

outputs:
two scalar rewards

loss: increase difference of predicted reward

$$L_{\mathrm{PM}} = \log\left(1 + e^{r_{\mathrm{rejected}} - r_{\mathrm{chosen}}}\right)$$

Modeling the reward and training reward model:

Q: Human-in-the-loop is expensive!

Solution: Instead of asking humans directly, we train a separate reward model to learn human preferences.

$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

y_w: winning sample

y_l: losing sample

y_w should score higher than y_l

3. Training Policy in RLHF

We have the following:

- A pretrained (possibly instruction-finetuned) LM $\pi_{ref}(y|x)$
- A reward model $r_{\phi}(x, y)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons

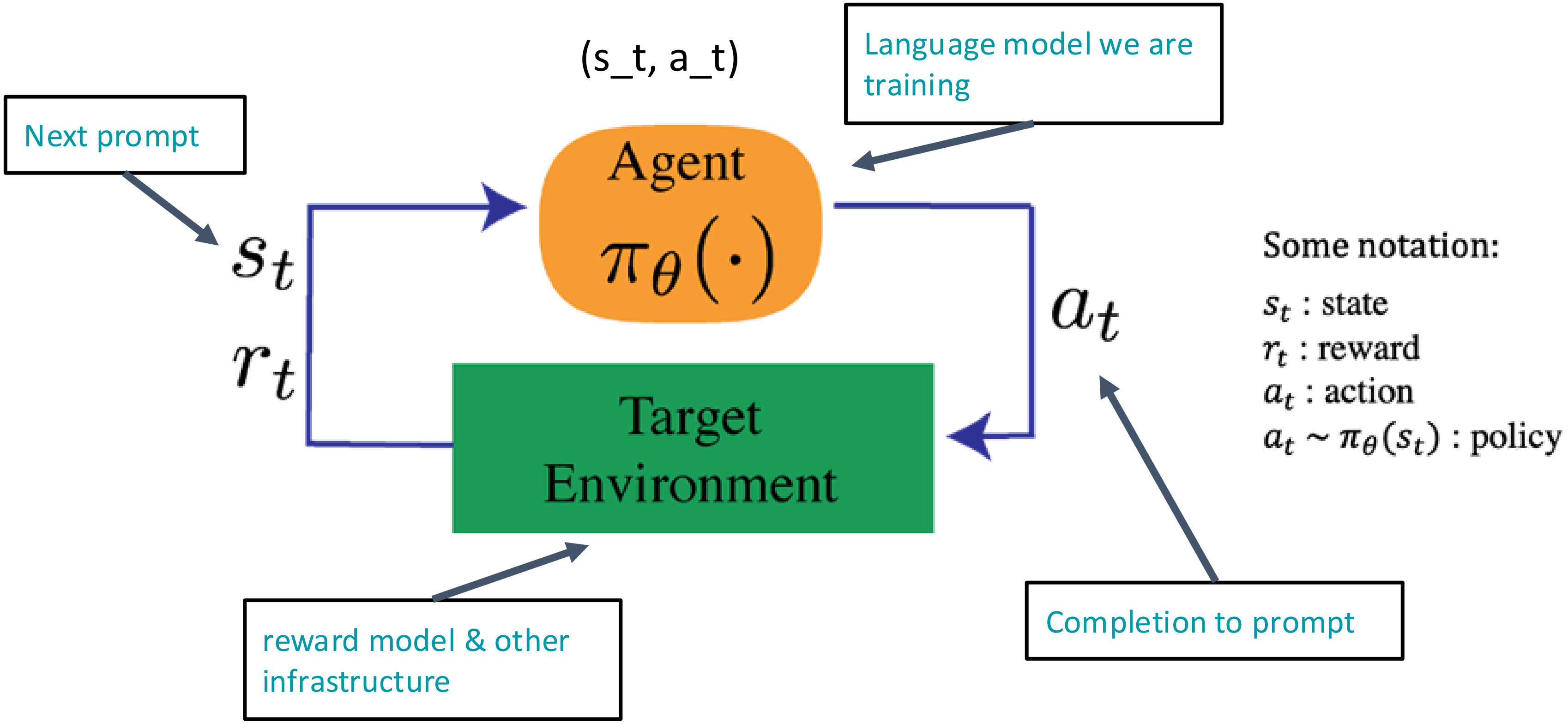
Now to do RLHF:

$$\pi_\theta^*(y|x) = \max_{\pi_\theta} \mathbb{E}_{x\sim D}\left[\mathbb{E}_{y\sim\pi_\theta(y|x)} r_\phi(x, y) - \beta D_{\mathrm{KL}}(\pi_\theta(y|x)||\pi_{\mathrm{ref}}(y|x))\right]$$

Maximizing rewards

Minimizing divergence between current policy and reference policy

3. Review: reinforcement learning basics in LLM



(s_t, a_t)

Language model we are training

Next prompt

reward model & other infrastructure

Completion to prompt

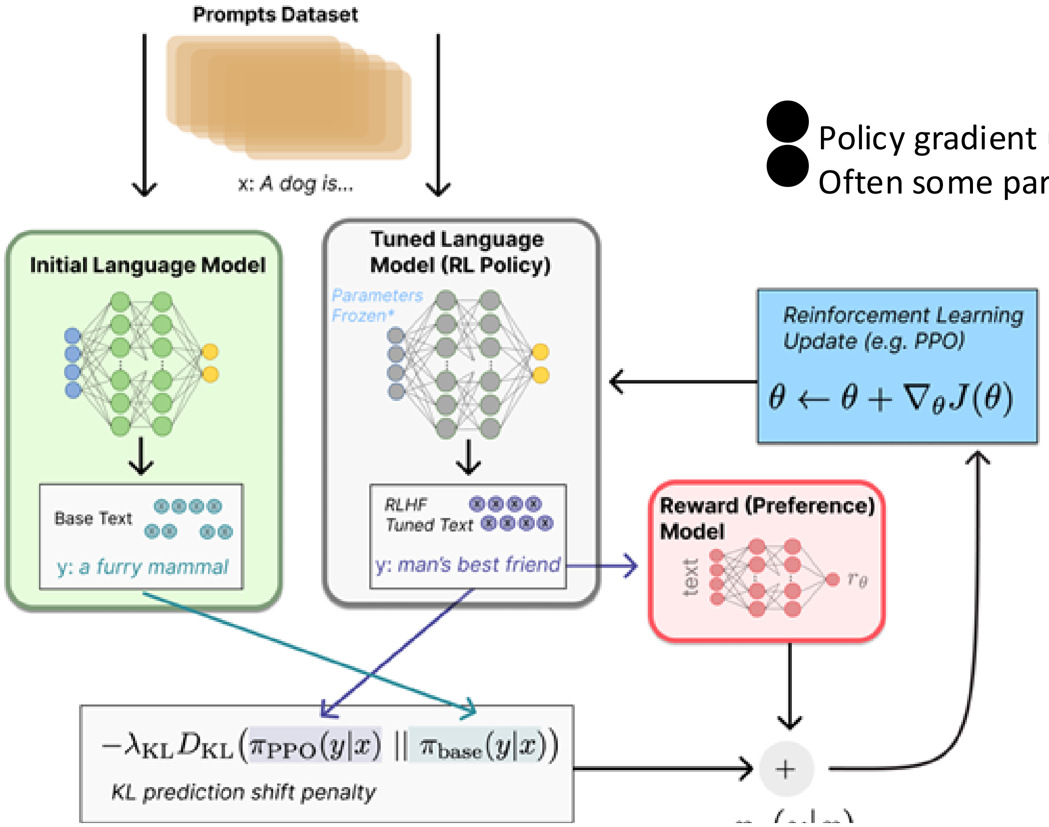
Some notation:

s_t : state
r_t : reward
a_t : action
$a_t \sim \pi_\theta(s_t)$: policy

RLHF at ICML2023, 12

Vs. Instruction Tuning- fine training with (x_t, y_t)

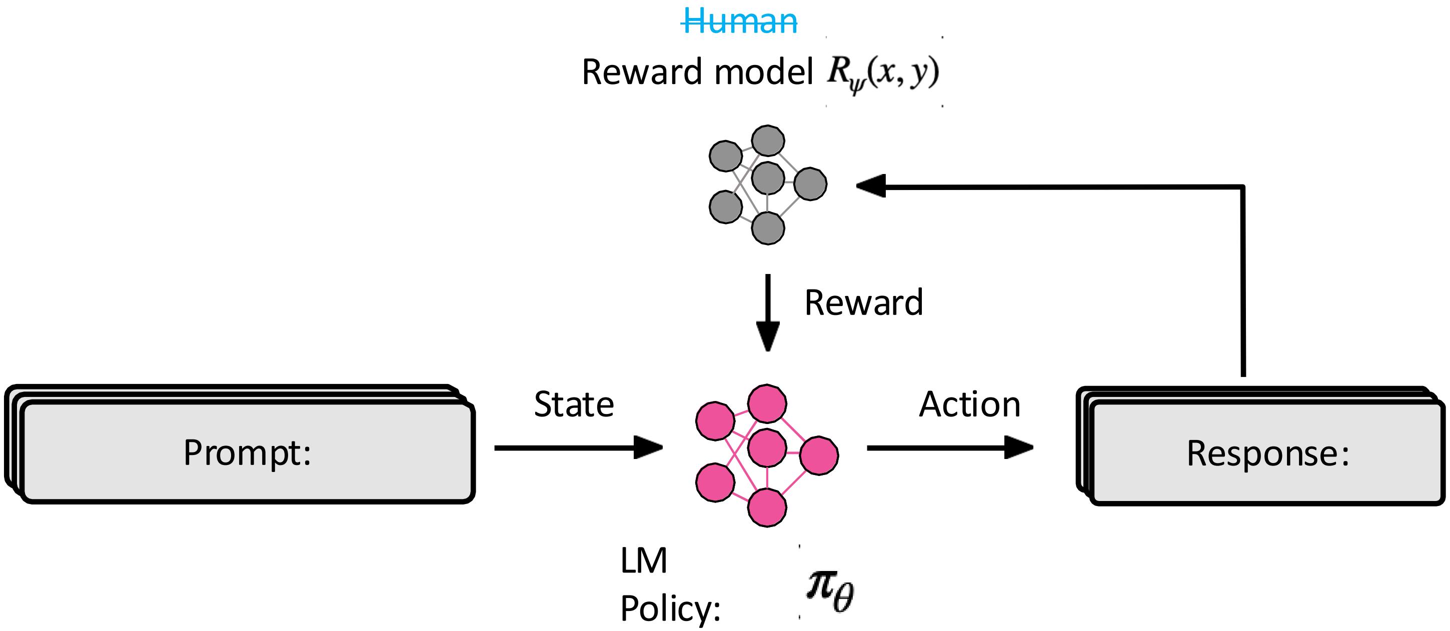
Fine tuning with RL – Reward feedback & Policy training


Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{\mathrm{KL}} D_{\mathrm{KL}}(\pi_{\mathrm{PPO}}(y|x) \,||\, \pi_{\mathrm{base}}(y|x))$
KL prediction shift penalty
+

- Policy gradient updates policy LM directly.
- Often some parameters of policy are frozen.

$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x) / \pi^{\mathrm{SFT}}(y \mid x)\right)\right]$$

Unpacking RLHF



~~Human~~

Reward model $R_\psi(x, y)$

Reward

State

Action

Prompt:

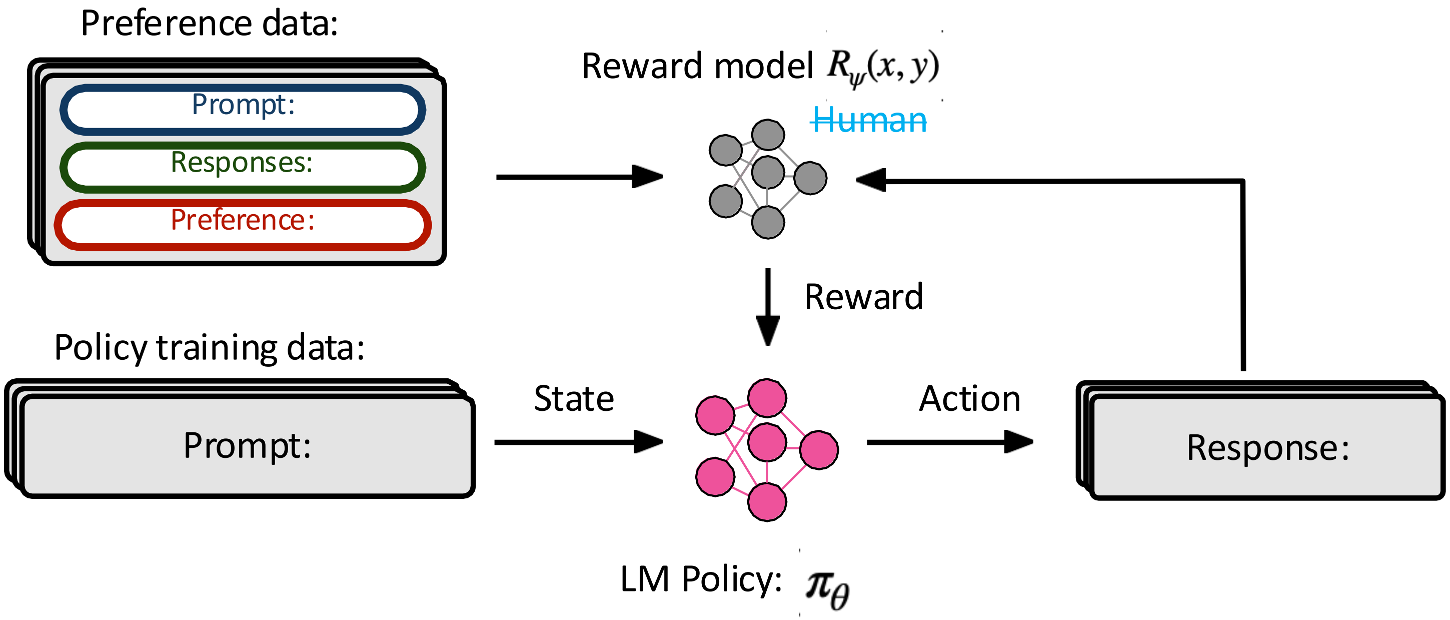
Response:

LM

Policy: π_θ

Credit: Ai2

Unpacking RLHF



Preference data:

Prompt:

Responses:

Preference:

Reward model $R_{\psi}(x, y)$

~~Human~~

Reward

Policy training data:

Prompt:

State

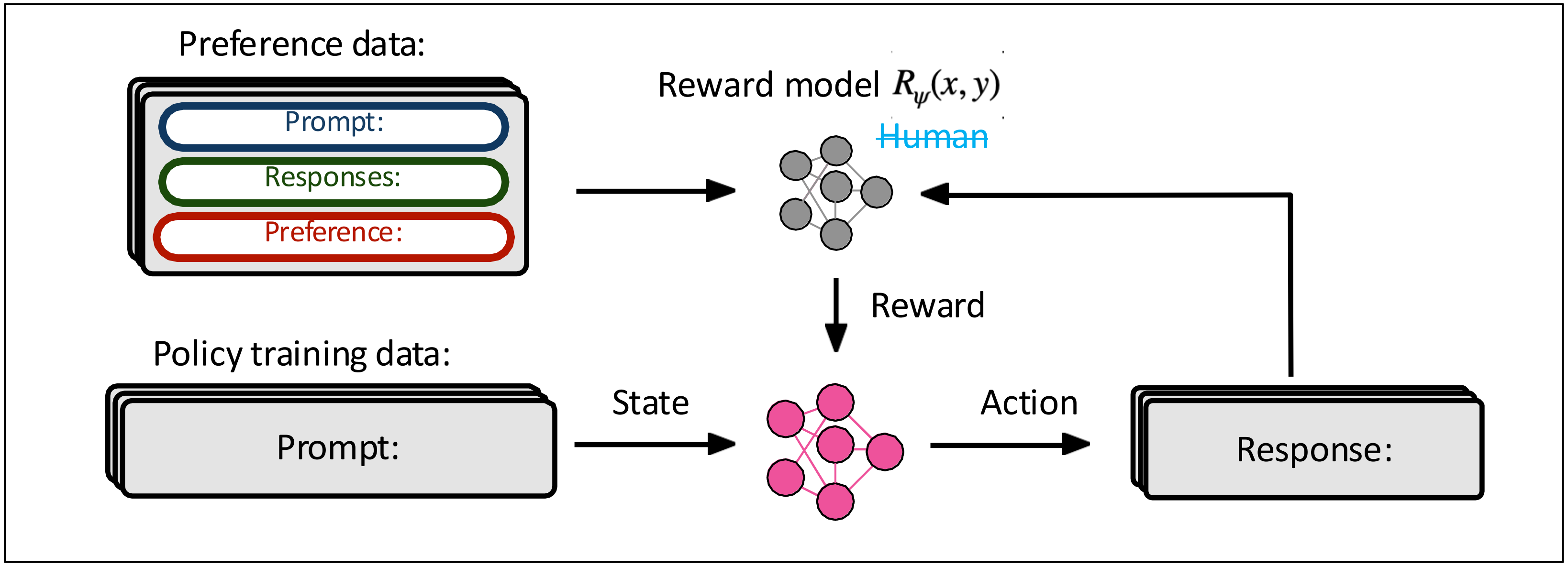
Action

Response:

LM Policy: π_{θ}

[Christiano et al., 2017]

Unpacking RLHF



Preference data:

Prompt:

Responses:

Preference:

Reward model $R_\psi(x, y)$

~~Human~~

Reward

Policy training data:

Prompt:

State

Action

Response:

PPO training:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}_\pi, y\sim\pi_\theta(y|x)}\Big[R_\psi(x, y)\Big] - \beta\mathbb{D}_{\mathrm{KL}}\big(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big)$$

[Shulman et al., 2017]

Dive Deep to
General RL: Proximal Policy Optimization (PPO)

Pseudocode

```
Initialize: policy parameters $\theta$
for k = 0, 1, 2 …
    collect set of completions $D_K$ from policy $\pi_K$
    compute reward of completions from reward model $r_K$
    compute value function (advantage) estimates
    update the policy parameters (PPO-Clip objective)
    update the value function (via gradient descent)
```

Generate from a LLM

Pass through reward model

Core RL part / math

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_{\phi}(s_t) - \hat{R}_t \right)^2$$

Spinning Up, Achiam 2018

DIVE DEEP TO PPO

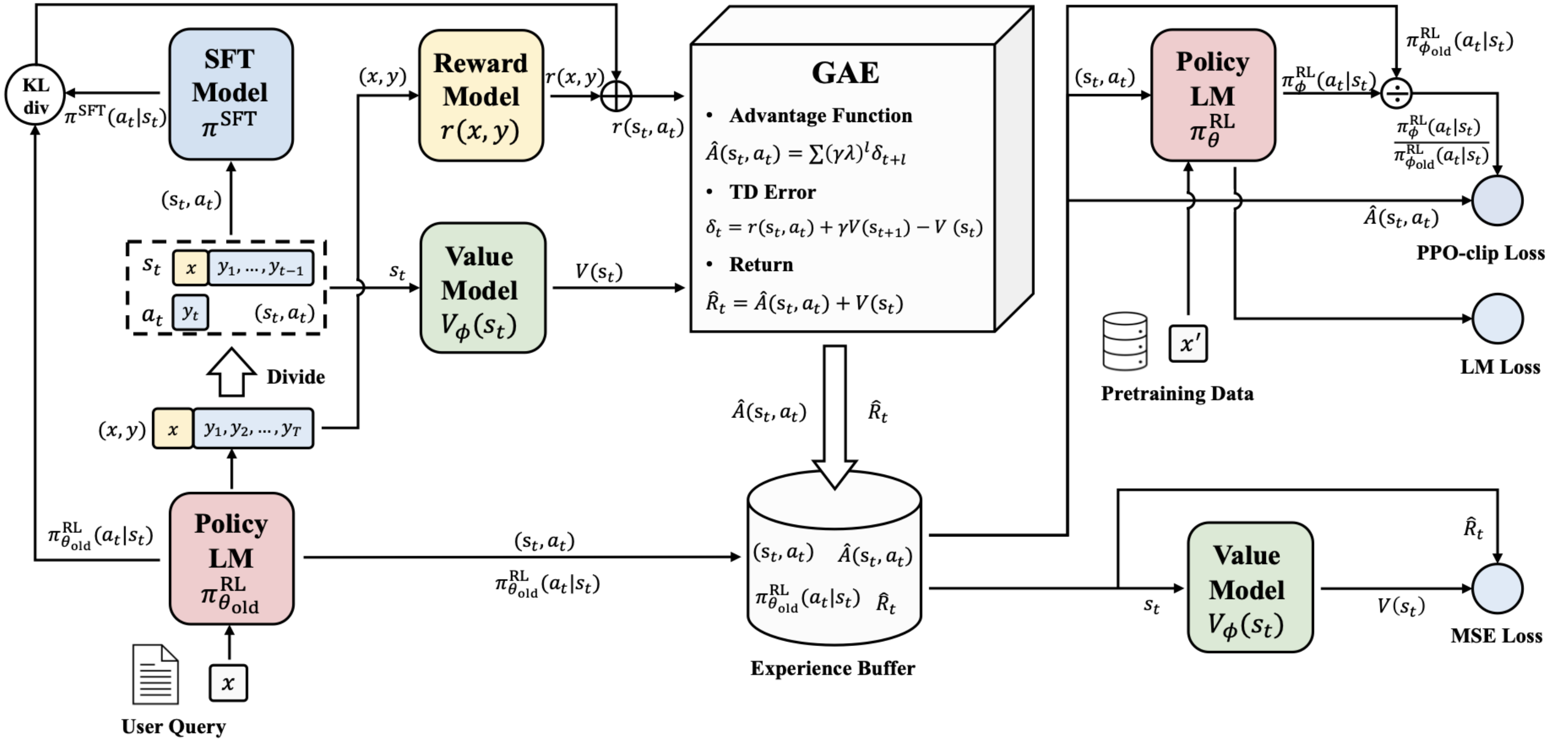
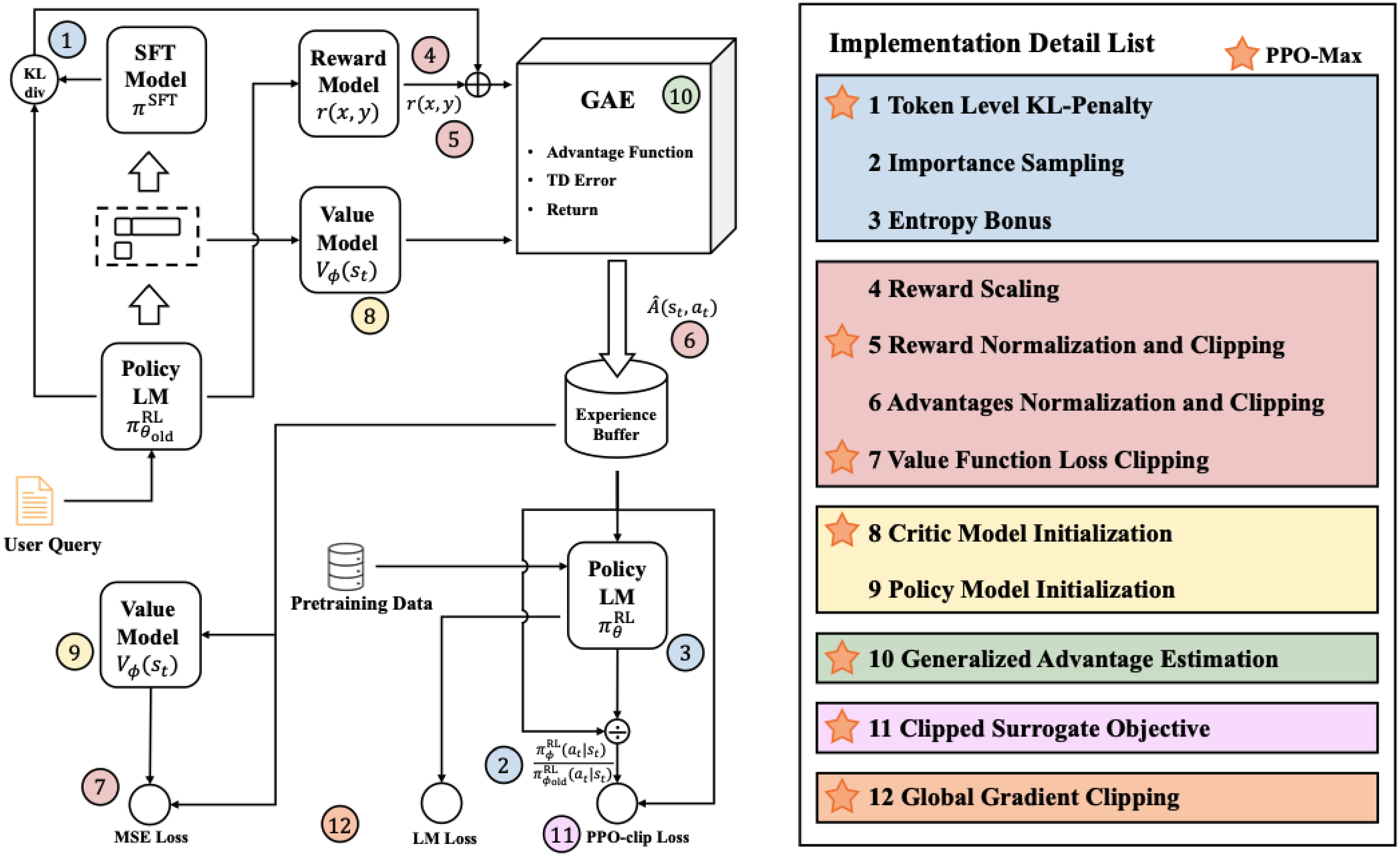

KL div
$\pi^{\text{SFT}}(a_t|s_t)$
SFT Model π^{SFT}
(x, y)
Reward Model $r(x, y)$
$r(x, y)$
$r(s_t, a_t)$
GAE
Advantage Function
$\hat{A}(s_t, a_t) = \sum(\gamma\lambda)^l \delta_{t+l}$
TD Error
$\delta_t = r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$
Return
$\hat{R}_t = \hat{A}(s_t, a_t) + V(s_t)$
(s_t, a_t)
s_t x $y_1, \ldots, y_{t-1}$
a_t y_t (s_t, a_t)
s_t
Value Model $V_\phi(s_t)$
$V(s_t)$
Divide
(x, y) x $y_1, y_2, \ldots, y_T$
$\pi^{\text{RL}}_{\theta_{\text{old}}}(a_t|s_t)$
Policy LM $\pi^{\text{RL}}_{\theta_{\text{old}}}$
(s_t, a_t)
$\pi^{\text{RL}}_{\theta_{\text{old}}}(a_t|s_t)$
x
User Query
$\hat{A}(s_t, a_t)$
$\hat{R}_t$
(s_t, a_t) $\hat{A}(s_t, a_t)$
$\pi^{\text{RL}}_{\theta_{\text{old}}}(a_t|s_t)$ $\hat{R}_t$
Experience Buffer
(s_t, a_t)
Policy LM π^{RL}_θ
$\pi^{\text{RL}}_\phi(a_t|s_t)$
$\pi^{\text{RL}}_{\phi_{\text{old}}}(a_t|s_t)$
$\frac{\pi^{\text{RL}}_\phi(a_t|s_t)}{\pi^{\text{RL}}_{\phi_{\text{old}}}(a_t|s_t)}$
$\hat{A}(s_t, a_t)$
PPO-clip Loss
x'
Pretraining Data
LM Loss
$\hat{R}_t$
s_t
Value Model $V_\phi(s_t)$
$V(s_t)$
MSE Loss

Figure 1: PPO workflow, depicting the sequential steps in the algorithm's execution. The process begins with sampling from the environment, followed by the application of GAE for improved advantage approximation. The diagram then illustrates the computation of various loss functions employed in PPO, signifying the iterative nature of the learning process and the policy updates derived from these losses.

https://arxiv.org/pdf/2307.04964 Secrets of RLHF in Large Language Models Part I: PPO

SFT Model: Supervised FineTuning Model; GAE: Generalized Advantage Estimation

DIVE DEEP TO PPO



Implementation Detail List ★ PPO-Max

- ★ 1 Token Level KL-Penalty
- 2 Importance Sampling
- 3 Entropy Bonus
- 4 Reward Scaling
- ★ 5 Reward Normalization and Clipping
- 6 Advantages Normalization and Clipping
- ★ 7 Value Function Loss Clipping
- ★ 8 Critic Model Initialization
- 9 Policy Model Initialization
- ★ 10 Generalized Advantage Estimation
- ★ 11 Clipped Surrogate Objective
- ★ 12 Global Gradient Clipping

Figure 5: **Left** shows an equivalent structure to the RLHF framework in Figure 1. **Right** shows an implementation detail list for PPO. The number with circle indicates where this strategy is used in the PPO training. The pentagram indicates the method used by PPO-max.

https://arxiv.org/pdf/2307.04964 Secrets of RLHF in Large Language Models Part I: PPO

SFT Model: Supervised FineTuning Model; GAE: Generalized Advantage Estimation

Issues of PPO

- **Problem:** Scaling RLHF training to larger models requires **efficiently allocating** at least **four component models (actor (policy model), critic(value model), reward, reference)** across multiple GPUs due to the memory limit of each accelerator.
- **Existing libraries:**
- Ray is a **distributed execution framework** that provides powerful scheduling and scaling capabilities for parallel and distributed computing workloads.
- vLLM is a fast and easy-to-use library for **LLM inference and serving.** It delivers state-of-the-art serving throughput through efficient management of attention key and value memory with **PagedAttention**, continuous batching of incoming requests, and fast model execution with CUDA graph.
- DeepSpeed is an **optimization library** designed to enhance the efficiency of **large-scale** deep-learning models.

PPO Implementation in OpenRLHF
Scheduling Optimization

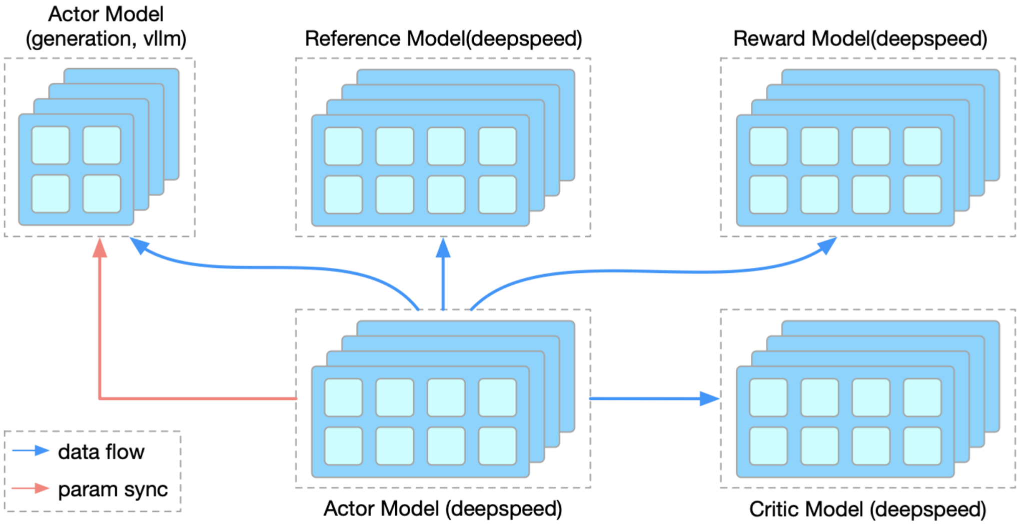
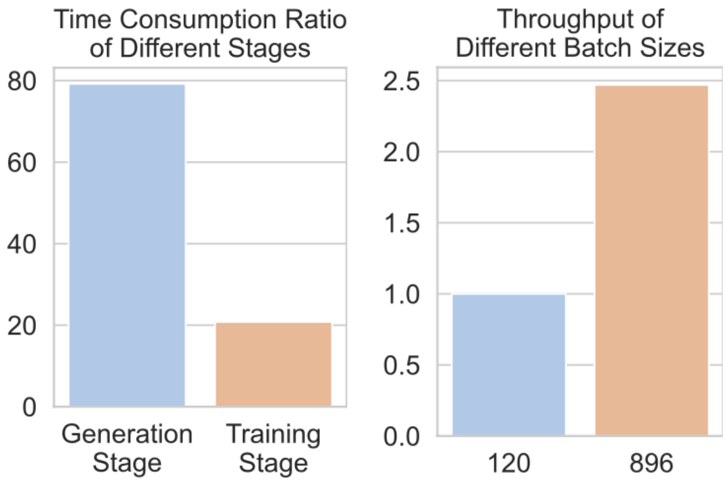


Figure 1: Ray Architecture of OpenRLHF. The four models in RLHF are distributed across different GPUs by Ray, which can also be freely merged or offloaded to save GPUs. The vLLM is used to accelerate actor generation. OpenRLHF synchronizes the weights of the ZeRO engine to the vLLM engine using the NVIDIA Collective Communications Library (NCCL).

PPO Performance Optimization



(a) RLHF stages (b) Infer Batch Size

Figure 4: Performance Profiling using LLaMA2 7B and NVIDIA A100.

- **The primary bottleneck** is at the **PPO sample generation** stage which takes up 80% of overall training time.
- Figure 4b shows that the **larger inference batch size** can significantly improve the generation throughput.
- OpenRLHF **distributes** the four models across multiple GPUs using Ray, effectively **increasing the batch size**.

Additional improvements:

- Offloading Adam optimizer states to the CPU frees up GPU memory, allowing for **larger batch sizes** during generation
- Employing **Flash Attention 2** accelerates Transformer model training.
- **Remove redundant padding** from training samples using PyTorch tensor slicing.

PPO Implementation Tricks

- Predict reward only on the end-of-text token of the sequence.
- Use token-level reinforcement learning for language models.
- Use Kullback–Leibler (KL) divergence loss term in PPO.
- Use pre-trained loss term in PPO, tuned based on a relative scale of the policy loss.
- Apply reward normalization for training stability.
- Apply distributed advantage normalization with global statistics.
- Use the Linear Warmup Cosine Annealing learning rate scheduler.
- Initialize the Critic with the weights of the reward model.
- Use a lower learning rate for the Actor while the Critic has a higher learning rate.
- Freeze the weights of the Actor in the initial learning stage for better initialization of the Critic.
- Use GAE (Generalized Advantage Estimation).

DeepSeek-R1: Incentivizes Reasoning In LLMs Through Reinforcement Learning

DeepSeek-AI Team

Presented by Meng Ma

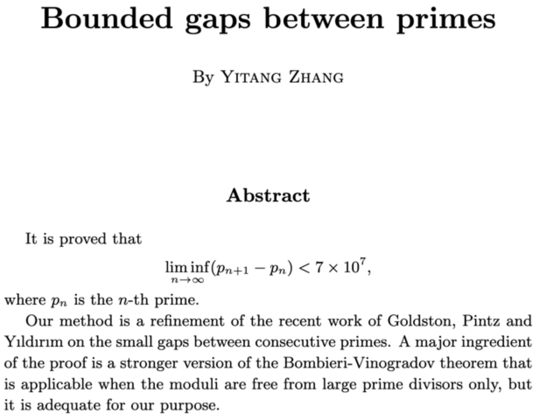
Feb 23, 2026

Reasoning matters for AI

Proving mathematical conjecture

Code generation and debugging

Planning and decision-making



Bounded gaps between primes

By Yitang Zhang

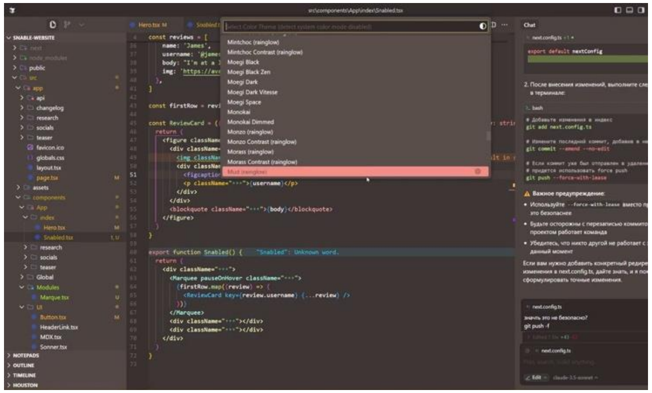
Abstract

It is proved that

$$\liminf_{n\to\infty}(p_{n+1}-p_n) < 7\times 10^7,$$

where p_n is the n-th prime.

Our method is a refinement of the recent work of Goldston, Pintz and Yıldırım on the small gaps between consecutive primes. A major ingredient of the proof is a stronger version of the Bombieri-Vinogradov theorem that is applicable when the moduli are free from large prime divisors only, but it is adequate for our purpose.



```
          <p className="···">{username}</p>
        </div>
      </div>
      <blockquote className="···">{body}</blockquote>
    </figure>
  )
}

export function Snabled() {    "Snabled": Unknown word.
  return (
    <div className="···">
      <Marquee pauseOnHover className="···">
        {firstRow.map((review) => (
          <ReviewCard key={review.username} {...review} />
        ))}
      </Marquee>
      <div className="···"></div>
      <div className="···"></div>
    </div>
  )
}
```

```
export default nextConfig
```

```
git add next.config.ts

git commit --amend --no-edit

# придется использовать force push
git push --force-with-lease
```

⚠ Важное предупреждение:

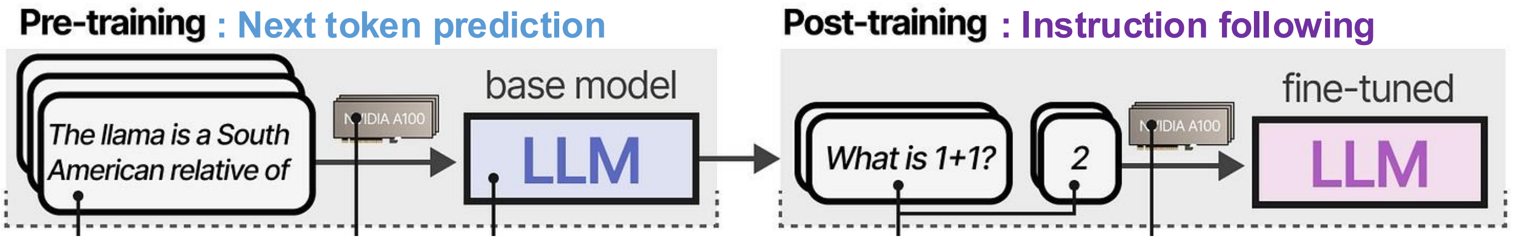
значить это не безопасно?
git push -f

Deductive reasoning

Abductive & Deductive reasoning

Causal reasoning

LLMs show signs of emergent reasoning capability

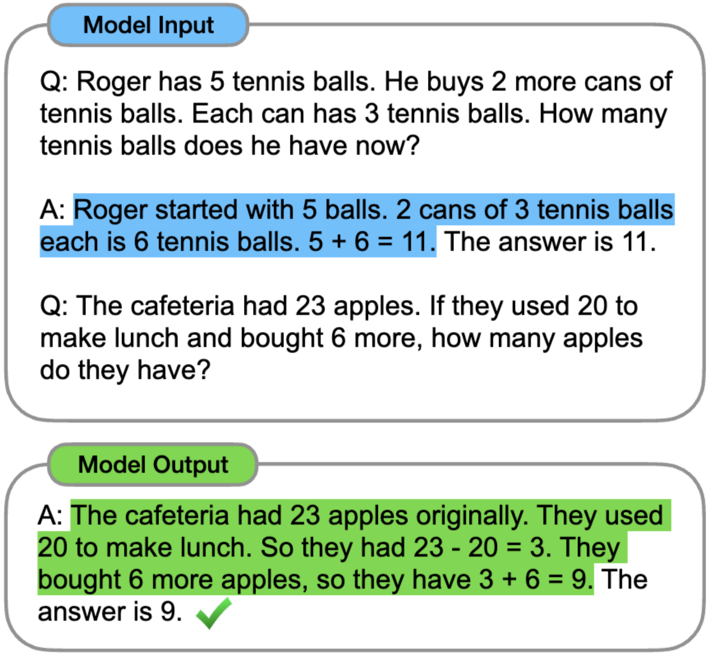

Pre-training : Next token prediction
The llama is a South American relative of
NVIDIA A100
base model
LLM
Post-training : Instruction following
What is 1+1?
2
NVIDIA A100
fine-tuned
LLM

Exam	GPT-4	GPT-4 (no vision)	GPT-3.5
Uniform Bar Exam (MBE+MEE+MPT)	298 / 400 (~90th)	298 / 400 (~90th)	213 / 400 (~10th)
LSAT	163 (~88th)	161 (~83rd)	149 (~40th)
SAT Evidence-Based Reading & Writing	710 / 800 (~93rd)	710 / 800 (~93rd)	670 / 800 (~87th)
SAT Math	700 / 800 (~89th)	690 / 800 (~89th)	590 / 800 (~70th)
Graduate Record Examination (GRE) Quantitative	163 / 170 (~80th)	157 / 170 (~62nd)	147 / 170 (~25th)
Graduate Record Examination (GRE) Verbal	169 / 170 (~99th)	165 / 170 (~96th)	154 / 170 (~63rd)
Graduate Record Examination (GRE) Writing	4 / 6 (~54th)	4 / 6 (~54th)	4 / 6 (~54th)
USABO Semifinal Exam 2020	87 / 150 (99th - 100th)	87 / 150 (99th - 100th)	43 / 150 (31st - 33rd)
USNCO Local Section Exam 2022	36 / 60	38 / 60	24 / 60

[GPT 4, OpenAI 2023]

LLMs perform strongly on many reasoning-heavy exams.

How can we improve LLM reasoning capability?



Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

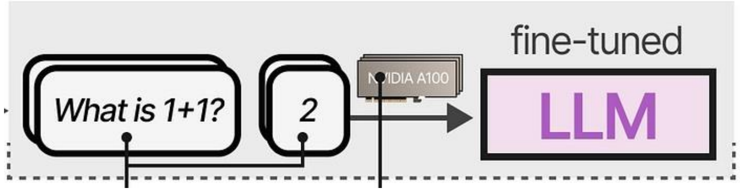
***In inference**: Chain-of-thought prompting*

"Human annotated reasoning data"

Input:
If a train travels 60 km in 1.5 hours, what is its average speed?

Let's think step by step.
1. The problem asks for the average speed of the train.
2. The formula is: Average Speed = Distance/Total Time.
3. Distance =60km, total time =1.5 hours
4. Speed = 60 km/ 1.5 h = 40 km/h.
Final Answer: The average speed of the train is 40 km/h.

Post-training : **SFT w/ reasoning data**



***In post-training**: Fine-tuning on reasoning data*

Limitations of existing approaches

Scalability Issue

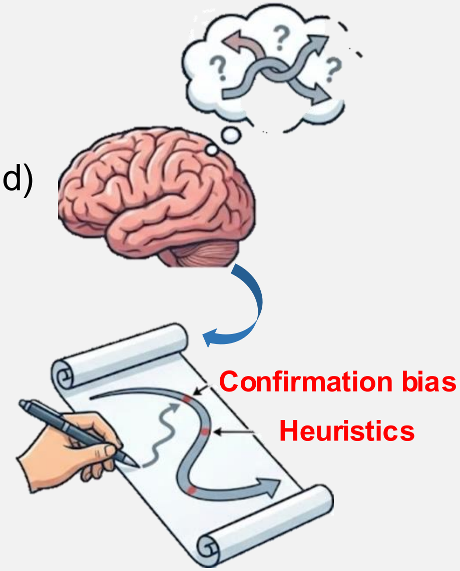
Vast unlabeled data
(No reasoning steps)

Human annotation
(Expansive & Slow)

Reasoning data
(Limited Scale)

Cognitive Bias

Human brain
(Subjective & Biased)



Human-written
reasoning trace

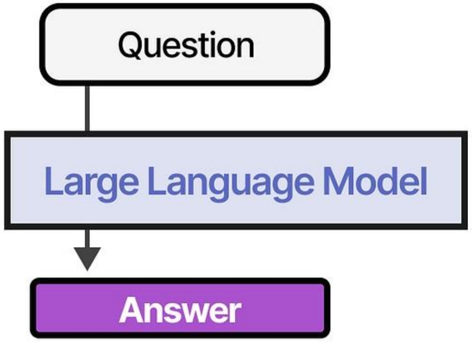
Capped by Human Reasoning

Constraining to mimic human thought processes may prevent exploring superior non-human-like reasonings.

How to elicit LLMs’ reasoning without heavily rely on human annotation?

DeepSeek-R1: Using pure RL to effectively elicit reasoning behavior

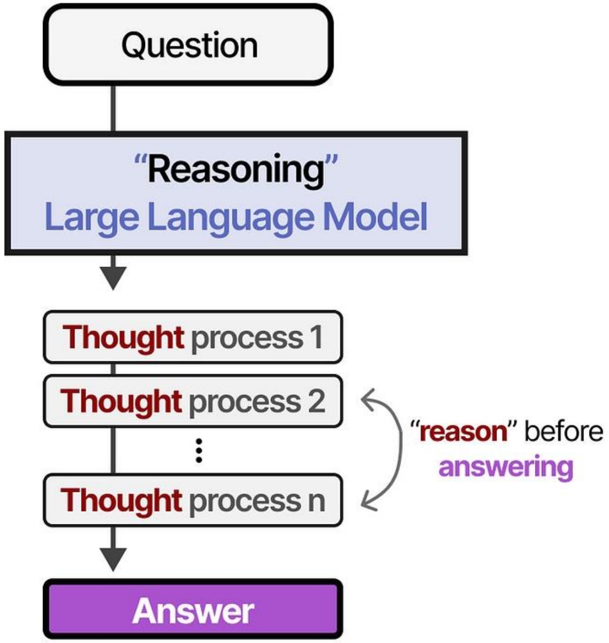
"Regular" LLMs


Question

Large Language Model

Answer

DeepSeek-R1 ("Reasoning" LLMs**)**


Question

"Reasoning"
Large Language Model

Thought process 1

Thought process 2

⋮

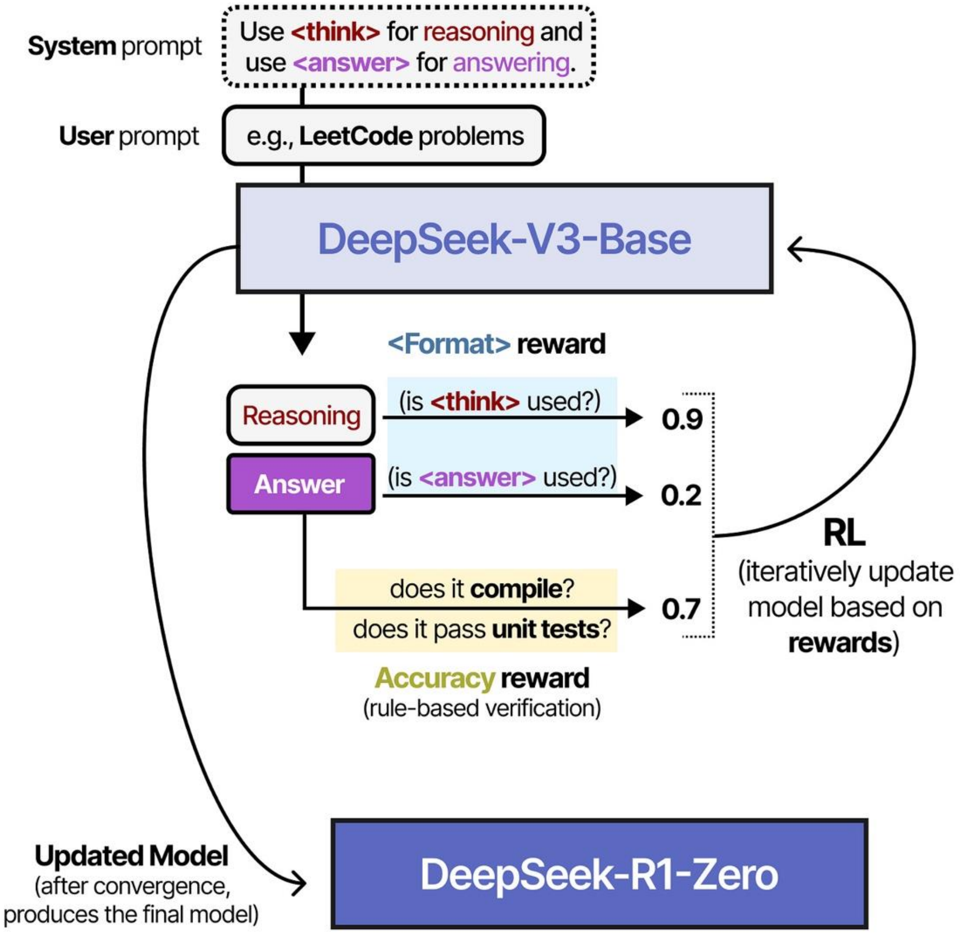
Thought process n

"reason" before answering

Answer

[Image credit: A Visual Guide to Reasoning LLMs]

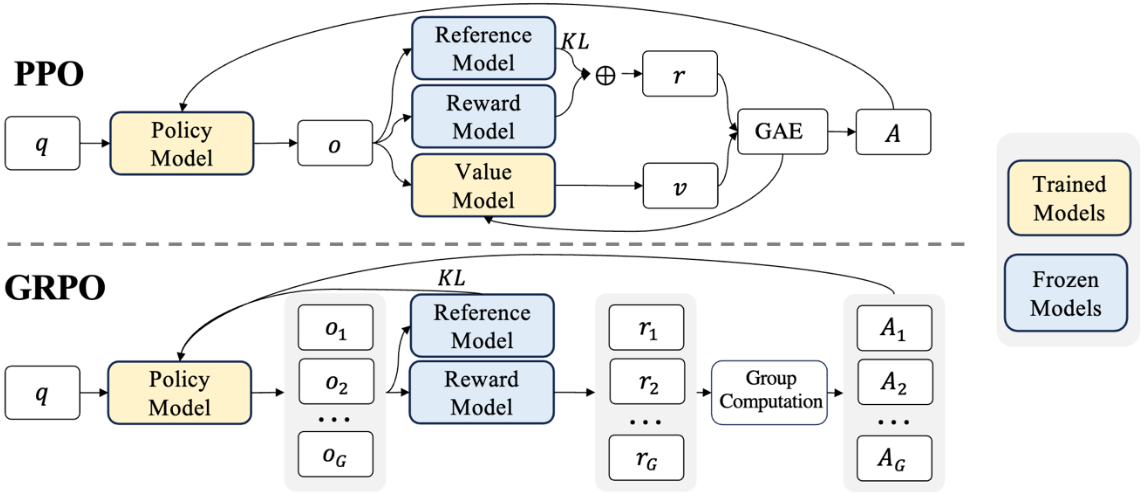
Reasoning with DeepSeek-R1 Zero



System prompt
Use **<think>** for reasoning and use **<answer>** for answering.
User prompt
e.g., **LeetCode** problems
DeepSeek-V3-Base
<Format> **reward**
Reasoning
Answer
(is **<think>** used?)
0.9
(is **<answer>** used?)
0.2
does it **compile**?
does it pass **unit tests**?
0.7
Accuracy reward
(rule-based verification)
RL
(iteratively update model based on **rewards**)
Updated Model
(after convergence, produces the final model)
DeepSeek-R1-Zero

[Image credit: A Visual Guide to Reasoning LLMs]

Reasoning with DeepSeek-R1 Zero



$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right),$$

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

Removal of the Value Model:

- PPO require a value model, while **GRPO remove the value model**.

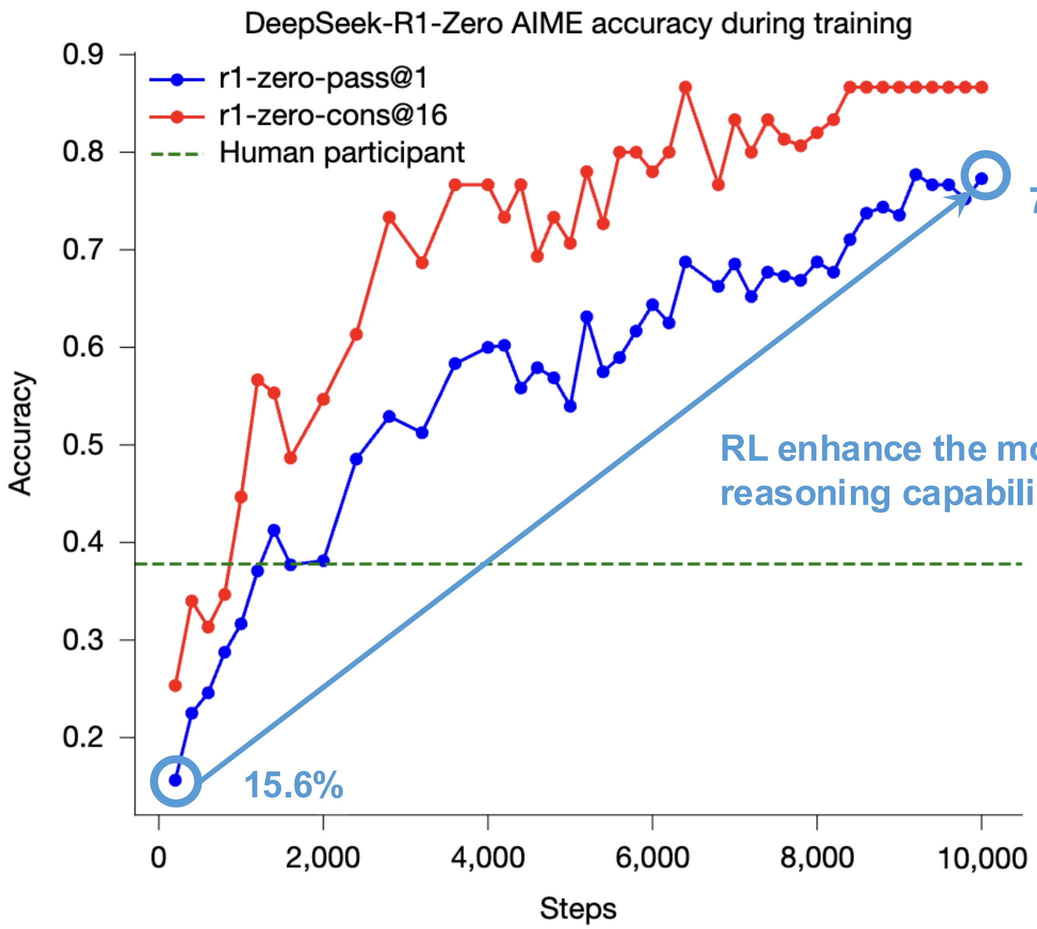
Individual vs. Group:

- PPO uses a Value Model and Generalized Advantage Estimation to compute a single advantage.
- **GRPO** uses Group Computation to **calculate** relative **advantages for a group of outputs**.

Efficiency:

- **GRPO** avoids training and running an additional large Value Model, it is much more **efficient for scaling up** the training of LLMs.

Performance of DeepSeek-R1 Zero

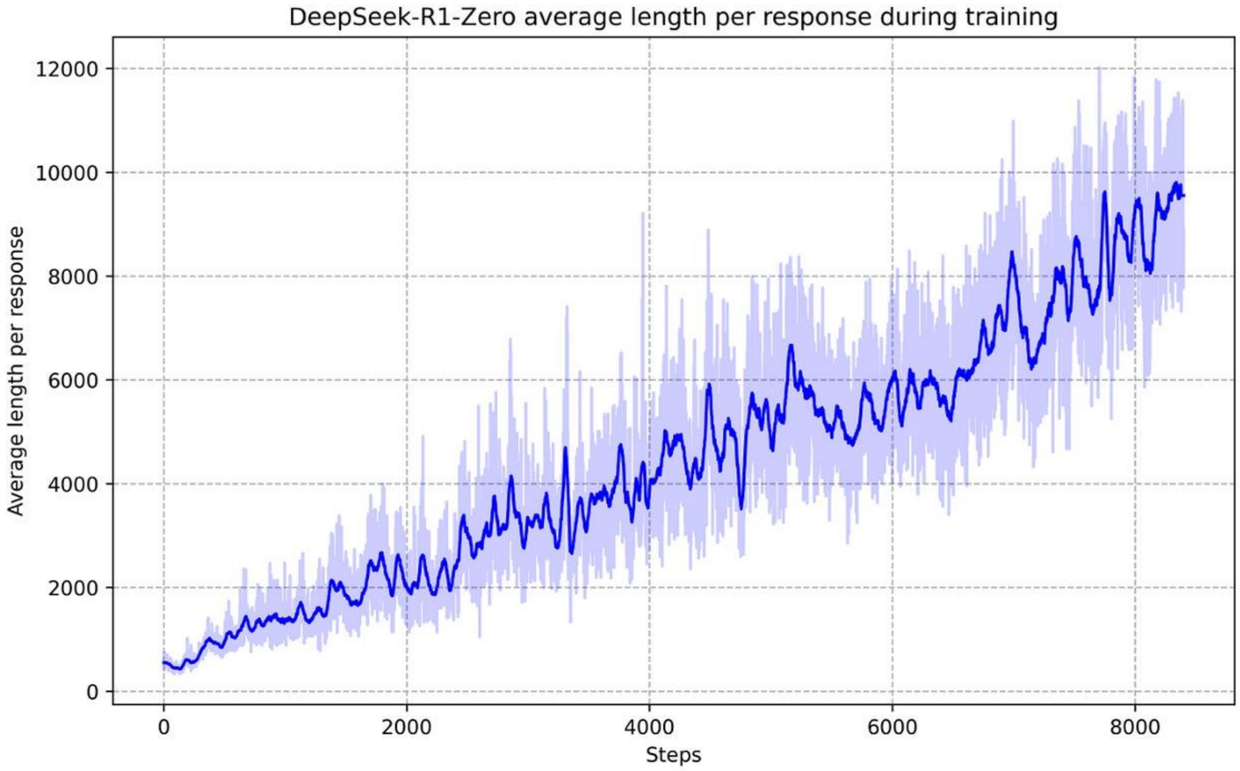


77.9%

RL enhance the model's reasoning capability.

15.6%

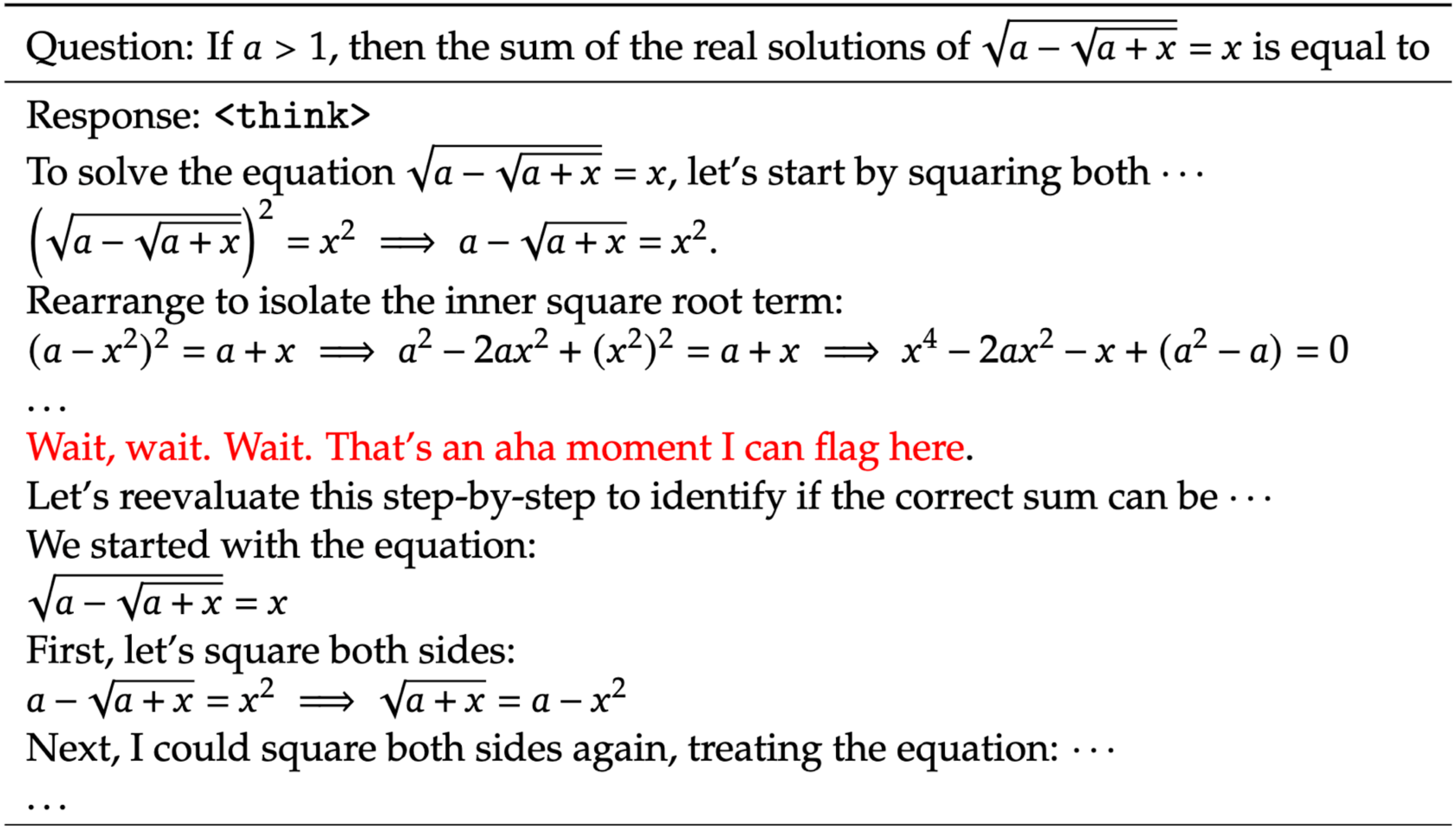
Performance of DeepSeek-R1 Zero



...the model learned to output **longer** and **longer reasoning** or **<think>** **responses.**

With each **training step...**

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:

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

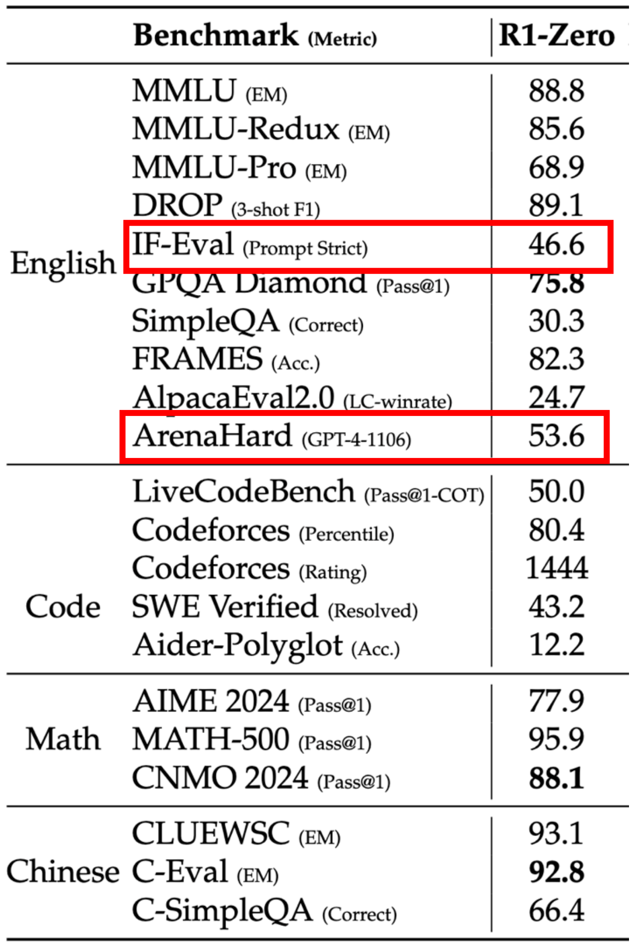
First, let's square both sides:

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

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

. . .

Issues of DeepSeek-R1 Zero

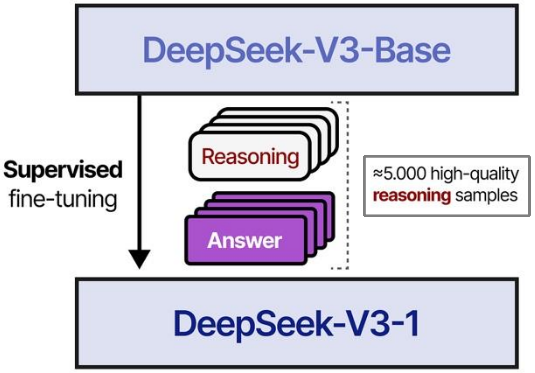


	Benchmark (Metric)	**R1-Zero**
English	MMLU (EM)	88.8
	MMLU-Redux (EM)	85.6
	MMLU-Pro (EM)	68.9
	DROP (3-shot F1)	89.1
	IF-Eval (Prompt Strict)	46.6
	GPQA Diamond (Pass@1)	75.8
	SimpleQA (Correct)	30.3
	FRAMES (Acc.)	82.3
	AlpacaEval2.0 (LC-winrate)	24.7
	ArenaHard (GPT-4-1106)	53.6
Code	LiveCodeBench (Pass@1-COT)	50.0
	Codeforces (Percentile)	80.4
	Codeforces (Rating)	1444
	SWE Verified (Resolved)	43.2
	Aider-Polyglot (Acc.)	12.2
Math	AIME 2024 (Pass@1)	77.9
	MATH-500 (Pass@1)	95.9
	CNMO 2024 (Pass@1)	**88.1**
Chinese	CLUEWSC (EM)	93.1
	C-Eval (EM)	**92.8**
	C-SimpleQA (Correct)	66.4

- **Poor readability**
- **Language mixing**

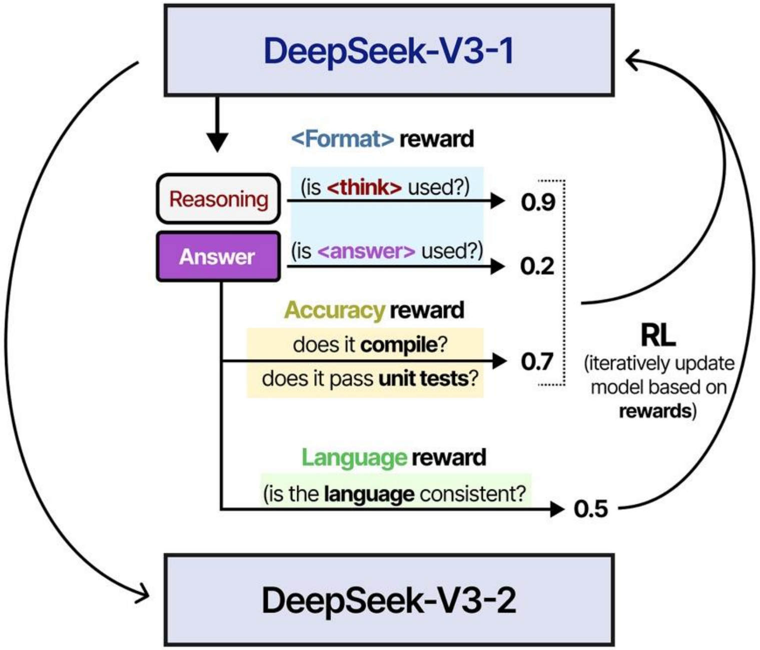
Reasoning with DeepSeek-R1

① Cold Start



Sampled from DeepSeek-R1 Zero

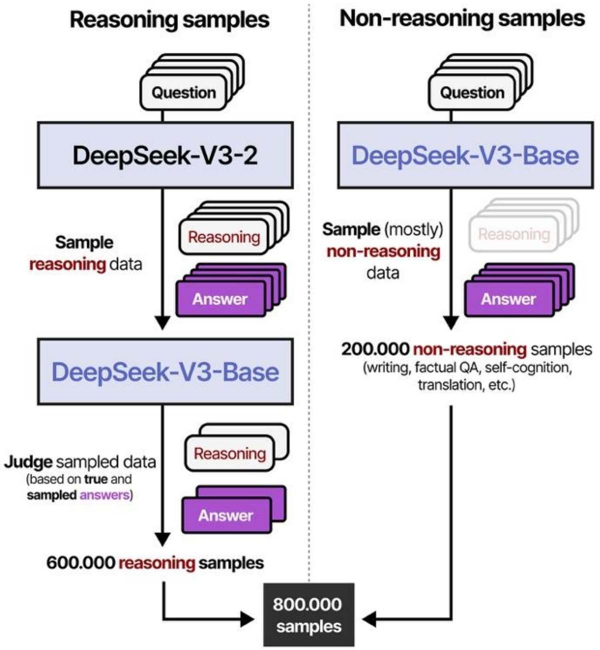
② Reasoning-oriented Reinforcement Learning



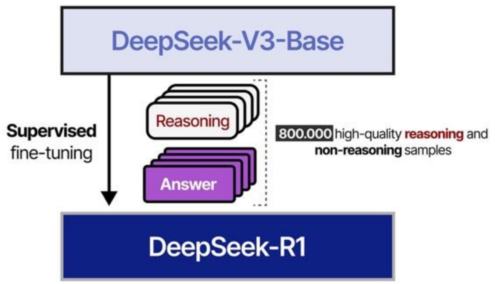
[Image credit: A Visual Guide to Reasoning LLMs]

Reasoning with DeepSeek-R1

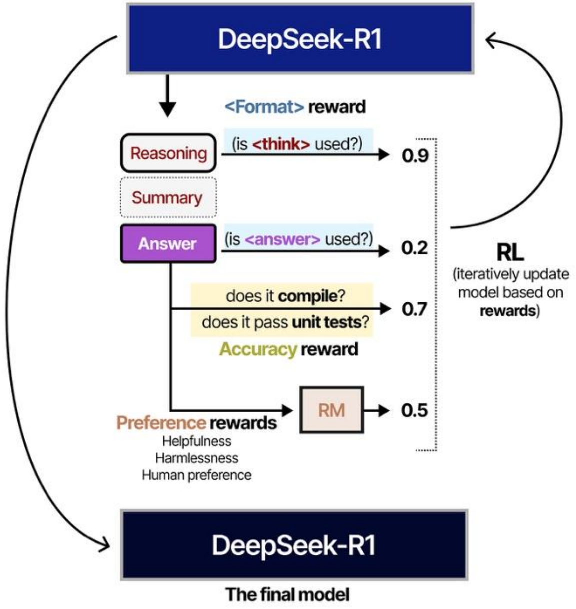
③ Rejection Sampling



④ Supervised Fine-Tuning

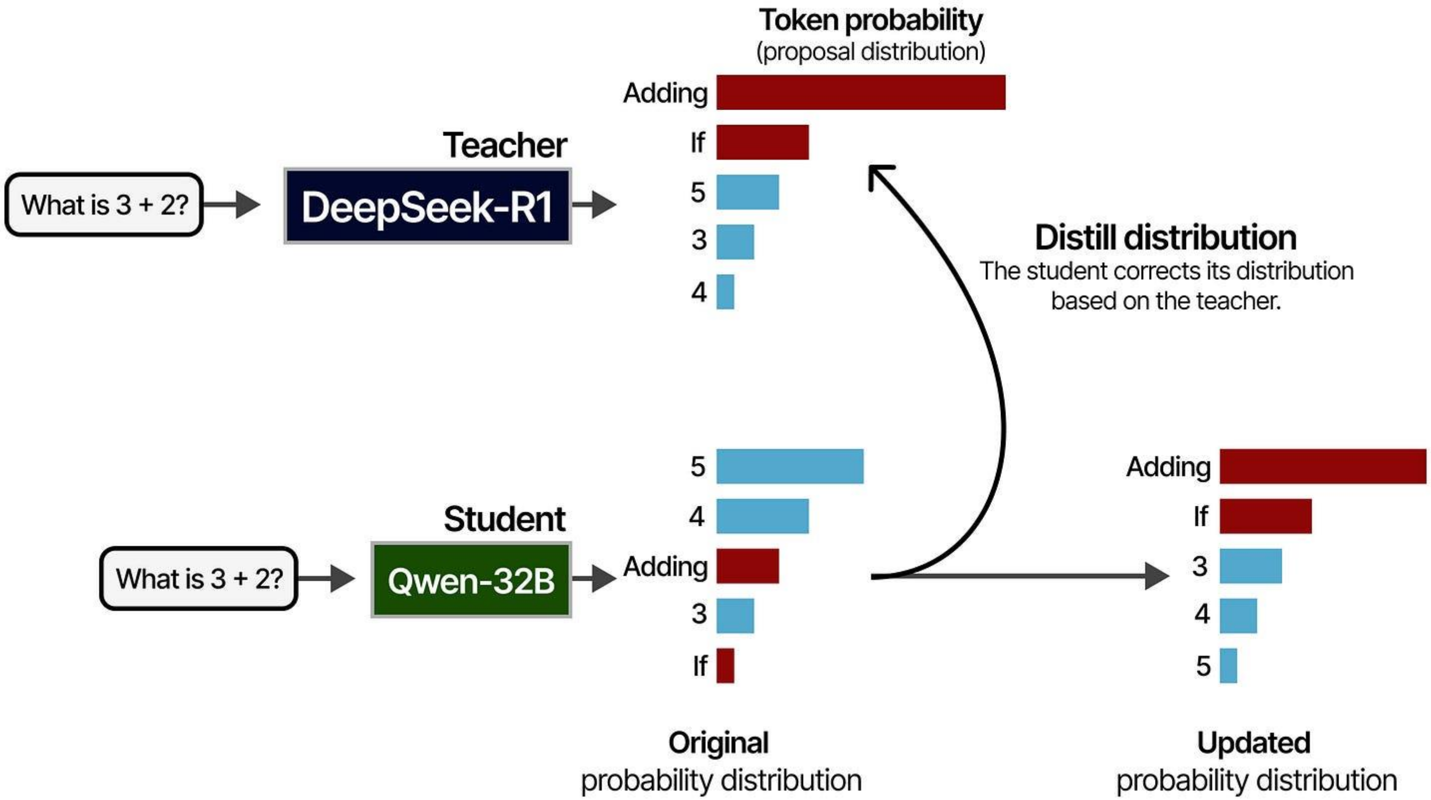


⑤ RL for all Scenarios



[Image credit: A Visual Guide to Reasoning LLMs]

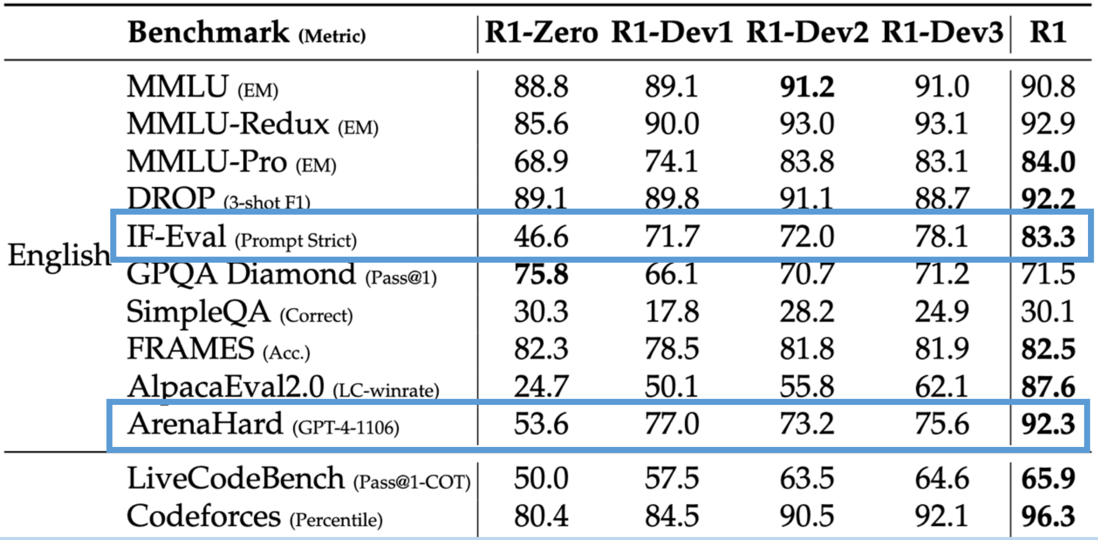
Distilling reasoning with DeepSeek-R1



DeepSeek-R1 serve as teacher model to distill reasoning to smaller student model.

[Image credit: A Visual Guide to Reasoning LLMs]

Performance of DeepSeek-R1



	Benchmark (Metric)	R1-Zero	R1-Dev1	R1-Dev2	R1-Dev3	R1
English	MMLU (EM)	88.8	89.1	**91.2**	91.0	90.8
	MMLU-Redux (EM)	85.6	90.0	93.0	93.1	92.9
	MMLU-Pro (EM)	68.9	74.1	83.8	83.1	**84.0**
	DROP (3-shot F1)	89.1	89.8	91.1	88.7	**92.2**
	IF-Eval (Prompt Strict)	46.6	71.7	72.0	78.1	**83.3**
	GPQA Diamond (Pass@1)	**75.8**	66.1	70.7	71.2	71.5
	SimpleQA (Correct)	30.3	17.8	28.2	24.9	30.1
	FRAMES (Acc.)	82.3	78.5	81.8	81.9	**82.5**
	AlpacaEval2.0 (LC-winrate)	24.7	50.1	55.8	62.1	**87.6**
	ArenaHard (GPT-4-1106)	53.6	77.0	73.2	75.6	**92.3**
	LiveCodeBench (Pass@1-COT)	50.0	57.5	63.5	64.6	**65.9**
	Codeforces (Percentile)	80.4	84.5	90.5	92.1	**96.3**

DeepSeek-R1 shows significantly better instruction-following performance.

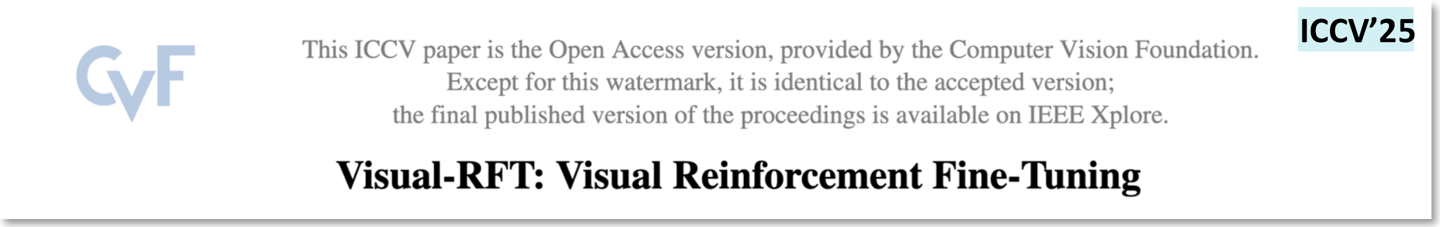
DeepSeek-R1's success:

RL + Verifiable Reward = LLM reasoning.

Does the same recipe hold for Multimodal LLMs?

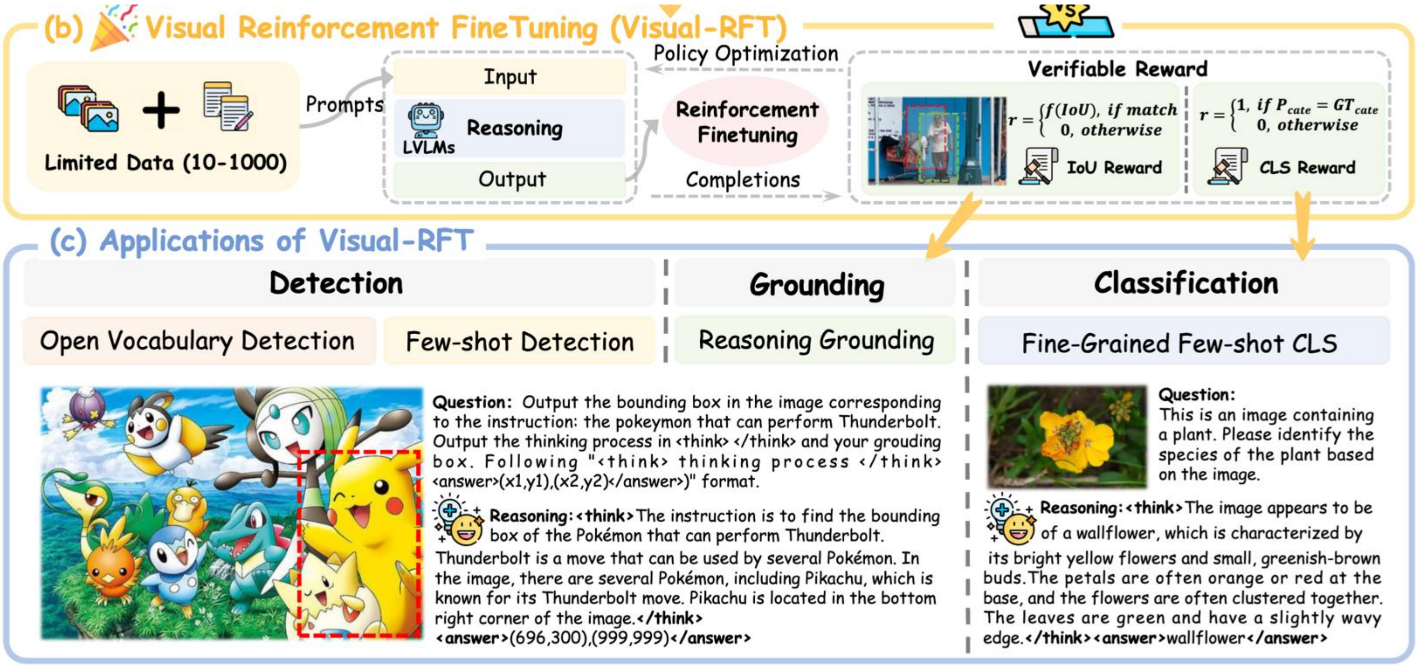
Reasoning with Multimodal LLMs

ICCV'25



This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Visual-RFT: Visual Reinforcement Fine-Tuning


(b) Visual Reinforcement FineTuning (Visual-RFT)
Limited Data (10-1000)
Prompts
Input
Reasoning
LVLMs
Output
Policy Optimization
Reinforcement Finetuning
Completions
Verifiable Reward
$r = \begin{cases} f(IoU), \text{ if match} \\ 0, \text{ otherwise} \end{cases}$
IoU Reward
$r = \begin{cases} 1, \text{ if } P_{cate} = GT_{cate} \\ 0, \text{ otherwise} \end{cases}$
CLS Reward
(c) Applications of Visual-RFT
Detection
Open Vocabulary Detection
Few-shot Detection
Grounding
Reasoning Grounding
Classification
Fine-Grained Few-shot CLS

Question: Output the bounding box in the image corresponding to the instruction: the pokeymon that can perform Thunderbolt. Output the thinking process in <think> </think> and your grouding box. Following "<think> thinking process </think> <answer>(x1,y1),(x2,y2)</answer>)" format.

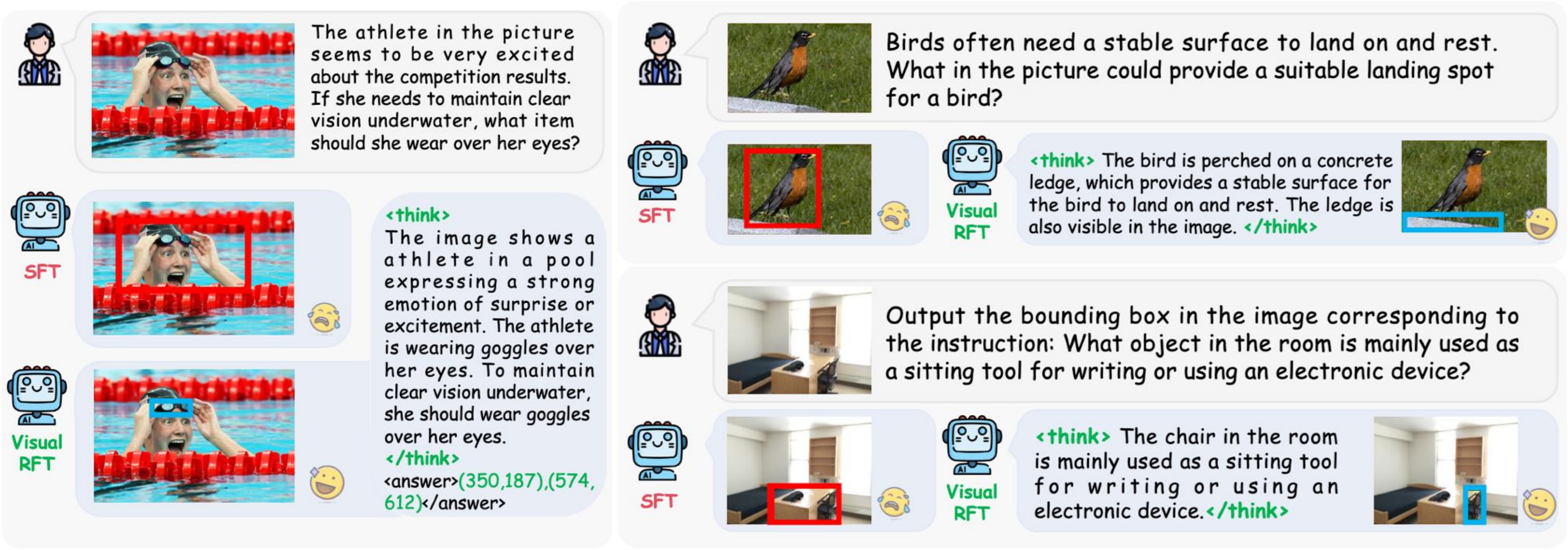
Reasoning:<think>The instruction is to find the bounding box of the Pokémon that can perform Thunderbolt. Thunderbolt is a move that can be used by several Pokémon. In the image, there are several Pokémon, including Pikachu, which is known for its Thunderbolt move. Pikachu is located in the bottom right corner of the image.**</think>**
<answer>(696,300),(999,999)**</answer>**

Question:
This is an image containing a plant. Please identify the species of the plant based on the image.

Reasoning:<think>The image appears to be of a wallflower, which is characterized by its bright yellow flowers and small, greenish-brown buds. The petals are often orange or red at the base, and the flowers are often clustered together. The leaves are green and have a slightly wavy edge.**</think><answer>**wallflower**</answer>**

Yes! RL + Verifiable reward enhance visual reasoning capability.

Reasoning with Multimodal LLMs



The athlete in the picture seems to be very excited about the competition results. If she needs to maintain clear vision underwater, what item should she wear over her eyes?

<think>
The image shows a athlete in a pool expressing a strong emotion of surprise or excitement. The athlete is wearing goggles over her eyes. To maintain clear vision underwater, she should wear goggles over her eyes.
</think>
<answer>(350,187),(574, 612)</answer>

Birds often need a stable surface to land on and rest. What in the picture could provide a suitable landing spot for a bird?

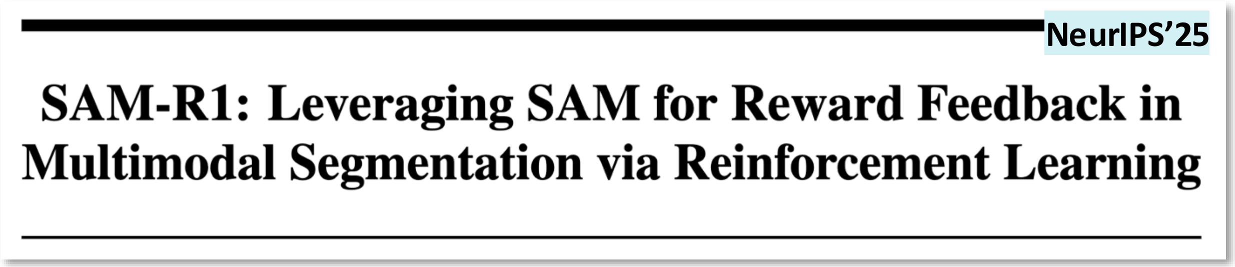
<think> The bird is perched on a concrete ledge, which provides a stable surface for the bird to land on and rest. The ledge is also visible in the image. </think>

Output the bounding box in the image corresponding to the instruction: What object in the room is mainly used as a sitting tool for writing or using an electronic device?

<think> The chair in the room is mainly used as a sitting tool for writing or using an electronic device.</think>

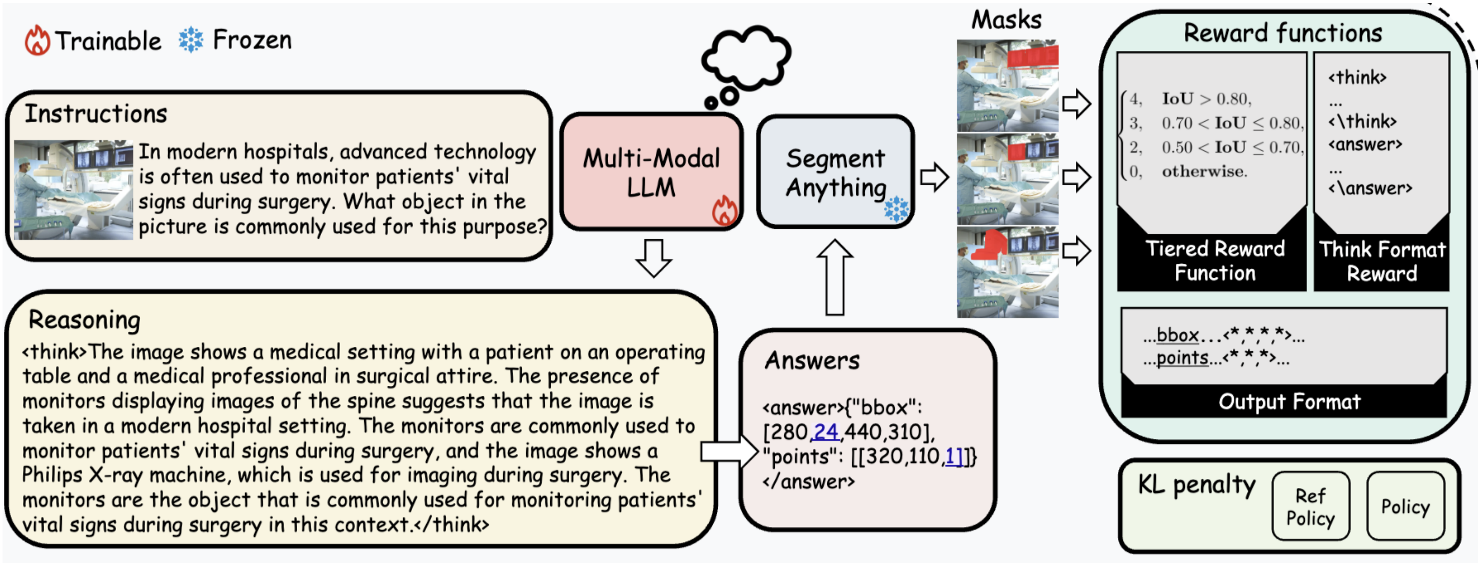
Thinking process significantly improves reasoning grounding ability with Visual-RFT.

Reasoning with Multimodal LLMs



NeurIPS'25

SAM-R1: Leveraging SAM for Reward Feedback in Multimodal Segmentation via Reinforcement Learning



Instructions: In modern hospitals, advanced technology is often used to monitor patients' vital signs during surgery. What object in the picture is commonly used for this purpose?

Reasoning: <think>The image shows a medical setting with a patient on an operating table and a medical professional in surgical attire. The presence of monitors displaying images of the spine suggests that the image is taken in a modern hospital setting. The monitors are commonly used to monitor patients' vital signs during surgery, and the image shows a Philips X-ray machine, which is used for imaging during surgery. The monitors are the object that is commonly used for monitoring patients' vital signs during surgery in this context.</think>

Answers:

```
<answer>{"bbox":
[280,24,440,310],
"points": [[320,110,1]]}
</answer>
```

Tiered Reward Function:

$$\begin{cases} 4, & \mathbf{IoU} > 0.80, \\ 3, & 0.70 < \mathbf{IoU} \leq 0.80, \\ 2, & 0.50 < \mathbf{IoU} \leq 0.70, \\ 0, & \textbf{otherwise}. \end{cases}$$

Think Format Reward:

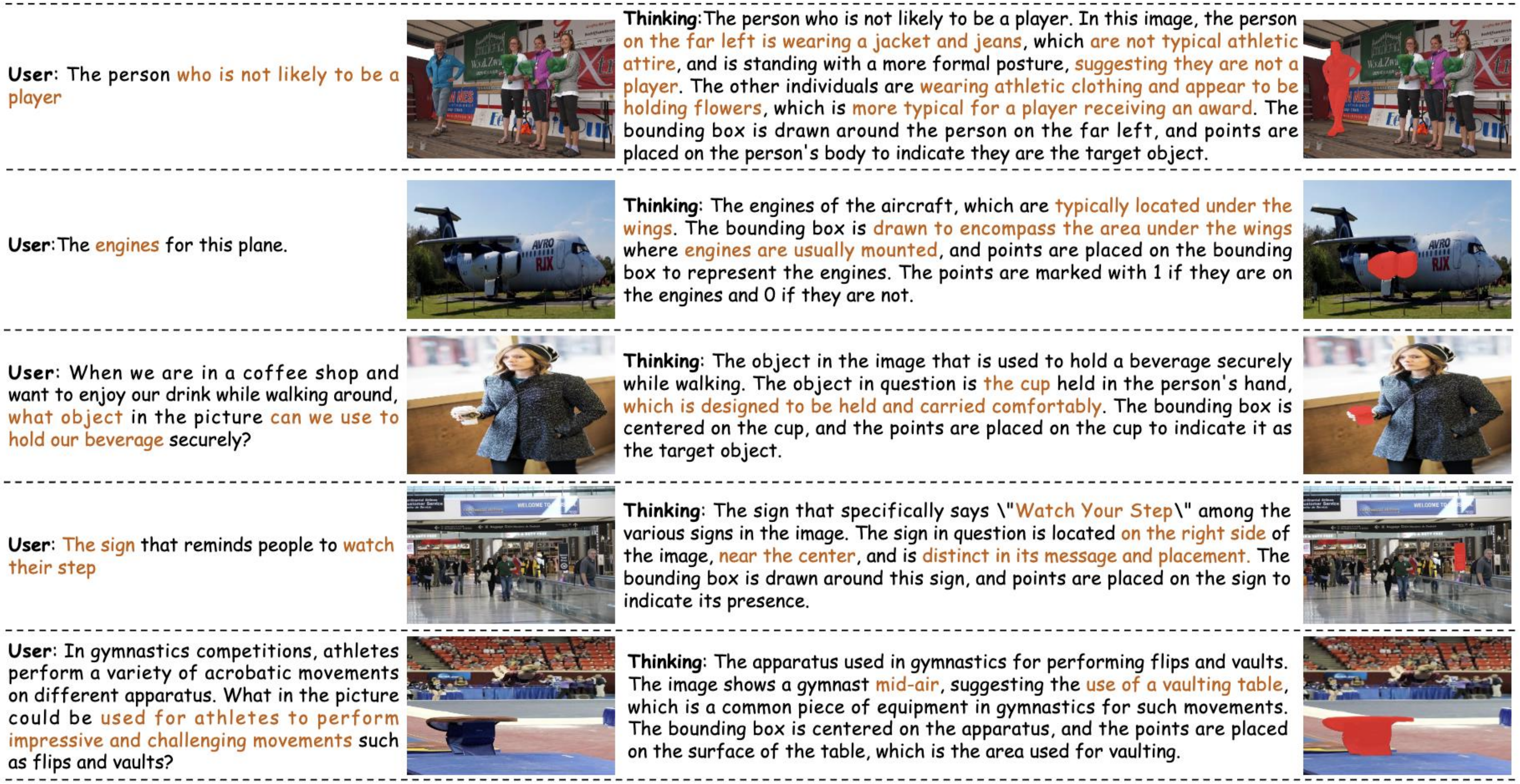
```
<think>
...
<\think>
<answer>
...
<\answer>
```

Output Format:

```
...bbox...<*,*,*,*>...
...points...<*,*,*>...
```

The same recipe also works for reasoning segmentation.

Reasoning with Multimodal LLMs



User: The person who is not likely to be a player

Thinking:The person who is not likely to be a player. In this image, the person on the far left is wearing a jacket and jeans, which are not typical athletic attire, and is standing with a more formal posture, suggesting they are not a player. The other individuals are wearing athletic clothing and appear to be holding flowers, which is more typical for a player receiving an award. The bounding box is drawn around the person on the far left, and points are placed on the person's body to indicate they are the target object.

User:The engines for this plane.

Thinking: The engines of the aircraft, which are typically located under the wings. The bounding box is drawn to encompass the area under the wings where engines are usually mounted, and points are placed on the bounding box to represent the engines. The points are marked with 1 if they are on the engines and 0 if they are not.

User: When we are in a coffee shop and want to enjoy our drink while walking around, what object in the picture can we use to hold our beverage securely?

Thinking: The object in the image that is used to hold a beverage securely while walking. The object in question is the cup held in the person's hand, which is designed to be held and carried comfortably. The bounding box is centered on the cup, and the points are placed on the cup to indicate it as the target object.

User: The sign that reminds people to watch their step

Thinking: The sign that specifically says \"Watch Your Step\" among the various signs in the image. The sign in question is located on the right side of the image, near the center, and is distinct in its message and placement. The bounding box is drawn around this sign, and points are placed on the sign to indicate its presence.

User: In gymnastics competitions, athletes perform a variety of acrobatic movements on different apparatus. What in the picture could be used for athletes to perform impressive and challenging movements such as flips and vaults?

Thinking: The apparatus used in gymnastics for performing flips and vaults. The image shows a gymnast mid-air, suggesting the use of a vaulting table, which is a common piece of equipment in gymnastics for such movements. The bounding box is centered on the apparatus, and the points are placed on the surface of the table, which is the area used for vaulting.

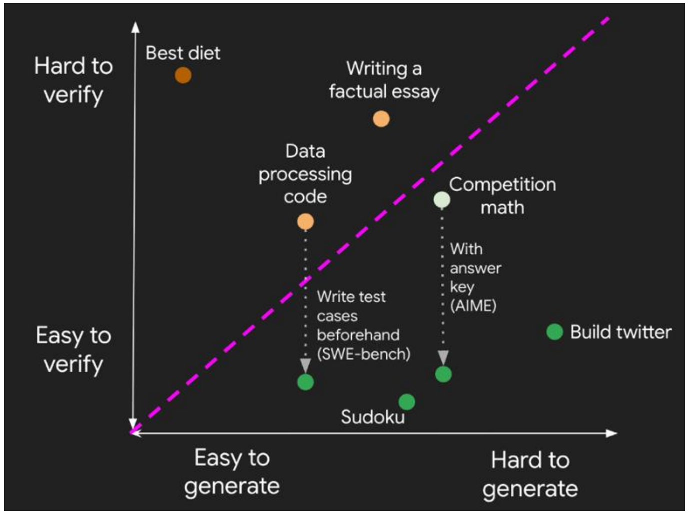
SAM-R1 shows strong zero-shot reasoning segmentation capability.

Takeaways

- RL can induce strong reasoning behavior.
- Ability to verify solutions is equivalent to ability to create an RL environment.

Verifier's rule: *The ease of training AI to solve a task is proportional to how verifiable the task is. All tasks that are possible to solve and easy to verify will be solved by AI.*

From Jason Wei (Author of the CoT Prompting)



AI solvable task has the following properties:

1. Objective truth: everyone agrees what good solutions are
2. Fast to verify: any given solution can be verified in a few seconds
3. Scalable to verify: many solutions can be verified simultaneously
4. Low noise: verification is as tightly correlated to the solution quality as possible
5. Continuous reward: it's easy to rank the goodness of many solutions for a single problem

A bit deep dive to GRPO

@Yanjun Qi

What is GRPO?

- Group Relative Policy Optimization (GRPO) is a **reinforcement learning** (RL) algorithm specifically designed to enhance reasoning capabilities in Large Language Models (LLMs). Unlike traditional RL methods, which rely heavily on external evaluators (critics) to guide learning, GRPO optimizes the model by evaluating **groups of responses** relative to one another. This approach enables more **efficient** training, making GRPO ideal for reasoning tasks that require complex problem-solving and long chains of thought.
- Proposed and used in DeepSeek R1

Why GRPO

- **Challenges** of Traditional RL methods like Proximal Policy Optimization (PPO)
- **Dependency on a Critic Model**:
  - PPO requires a separate critic model to estimate the value of each response, which doubles memory and computational requirements.
- **High Computational Cost**:
  - RL pipelines often demand significant computational resources to evaluate and optimize responses iteratively.
- **Scalability Issues**:
  - Absolute reward evaluations struggle with diverse tasks, making it hard to generalize across reasoning domains.

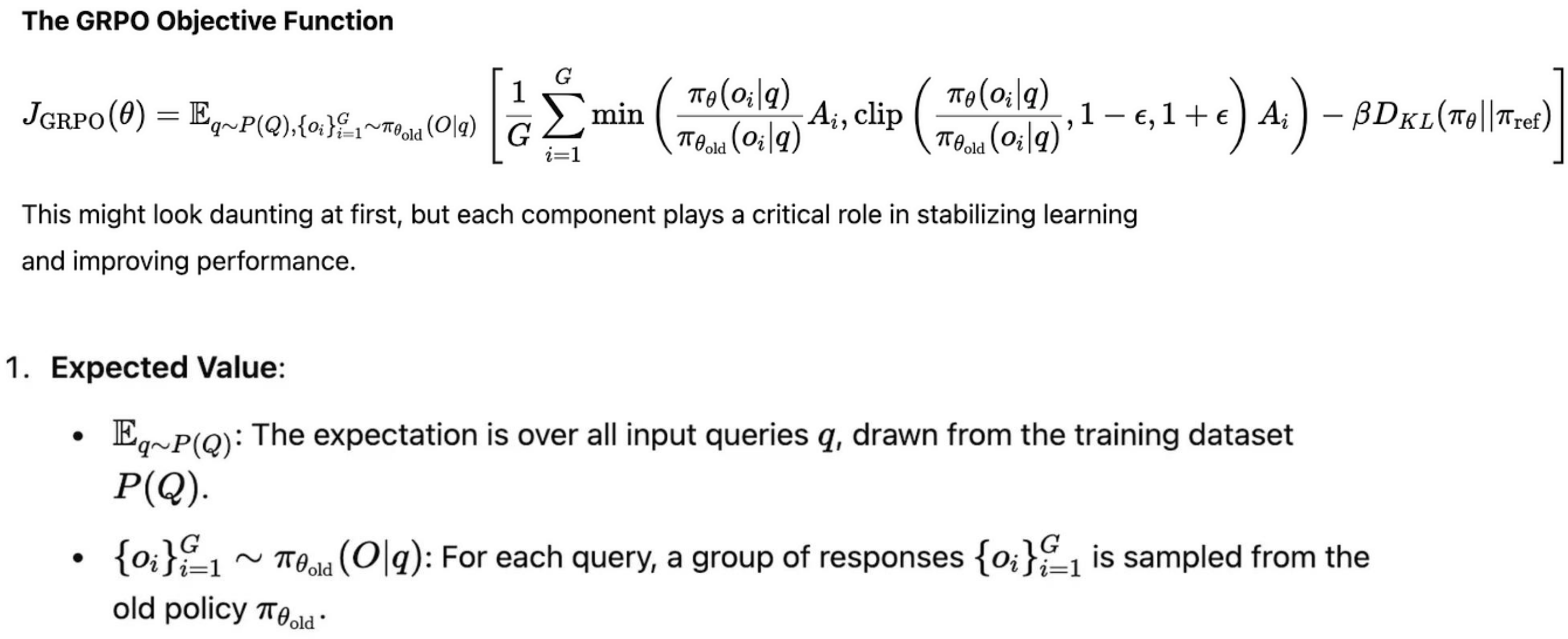
Why GRPO

- How GRPO **Addresses** These Challenges of PPO
- **Critic-Free Optimization**:
  - GRPO removes the need for a critic model by comparing responses within a group, significantly reducing computational overhead.
- **Relative Evaluation**:
  - Instead of relying on an external evaluator, GRPO uses group dynamics to assess how well a response performs relative to others in the same batch.
- **Efficient Training**:
  - By focusing on group-based advantages, GRPO simplifies the reward estimation process, making it faster and more scalable for large models.

Key Idea of GRPO: relative evaluation

- For each input query, the model generates a **group** of potential responses.
- These responses are scored based on how they **compare to others in the group**, rather than being evaluated in isolation.
- The advantage of a response reflects how much better or worse it is relative to the group's average performance.

Understanding the GRPO Objective Function



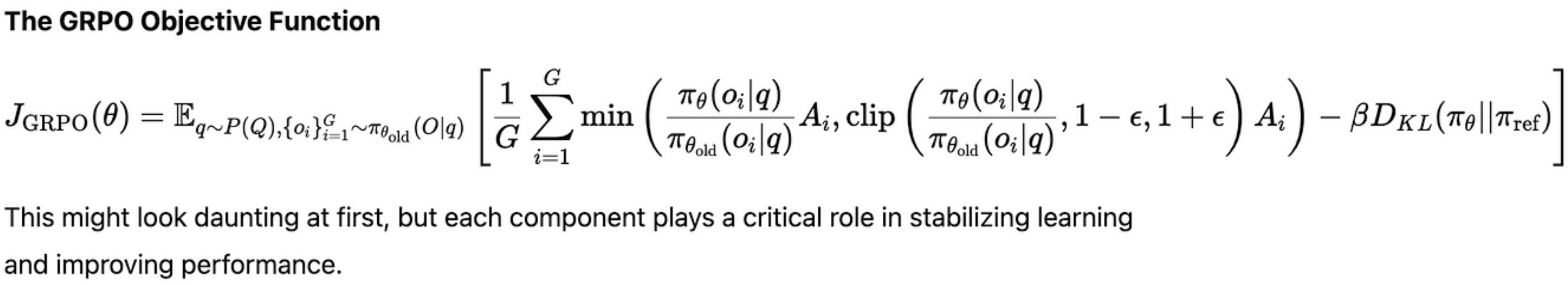
The GRPO Objective Function

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

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

1. **Expected Value**:
   - $\mathbb{E}_{q \sim P(Q)}$: The expectation is over all input queries q, drawn from the training dataset $P(Q)$.
   - $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O|q)$: For each query, a group of responses $\{o_i\}_{i=1}^{G}$ is sampled from the old policy $\pi_{\theta_{\text{old}}}$.

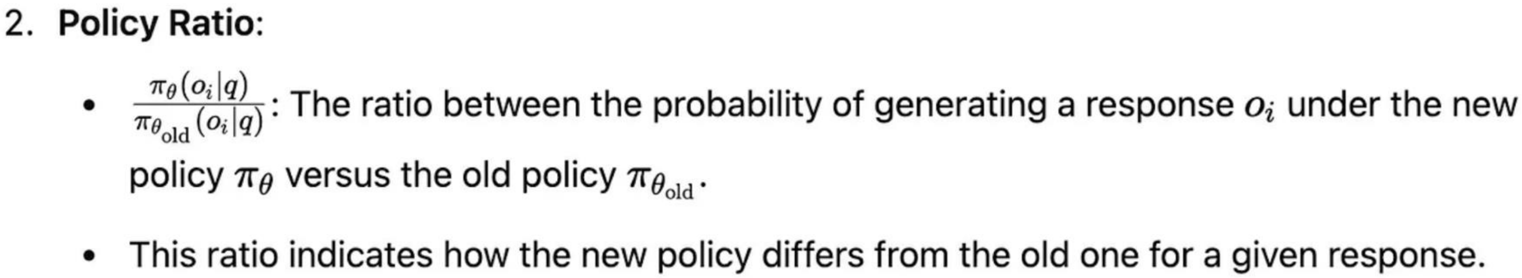
Understanding the GRPO Objective Function



The GRPO Objective Function

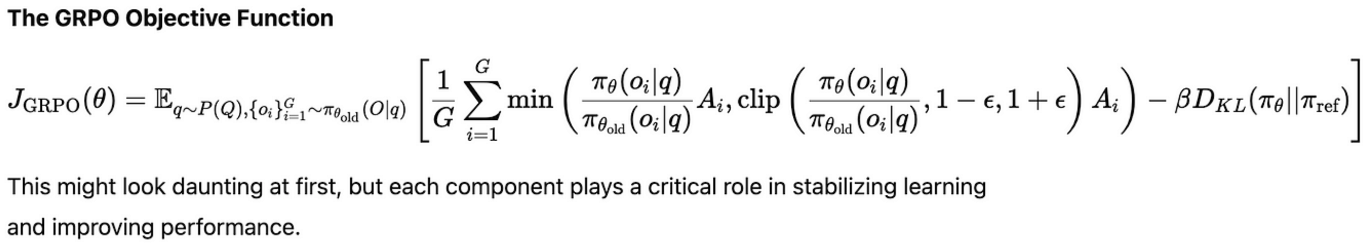
$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right)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

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.



2. **Policy Ratio:**
    - $\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$: The ratio between the probability of generating a response o_i under the new policy π_θ versus the old policy $\pi_{\theta_{\text{old}}}$.
    - This ratio indicates how the new policy differs from the old one for a given response.

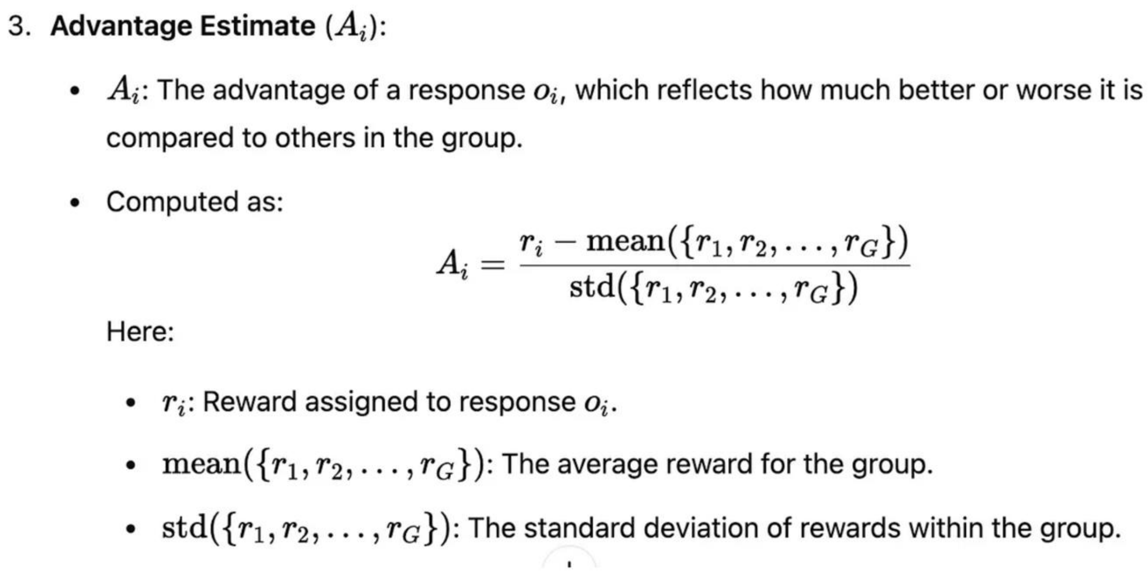
Understanding the GRPO Objective Function



The GRPO Objective Function

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

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.



3. **Advantage Estimate** (A_i):

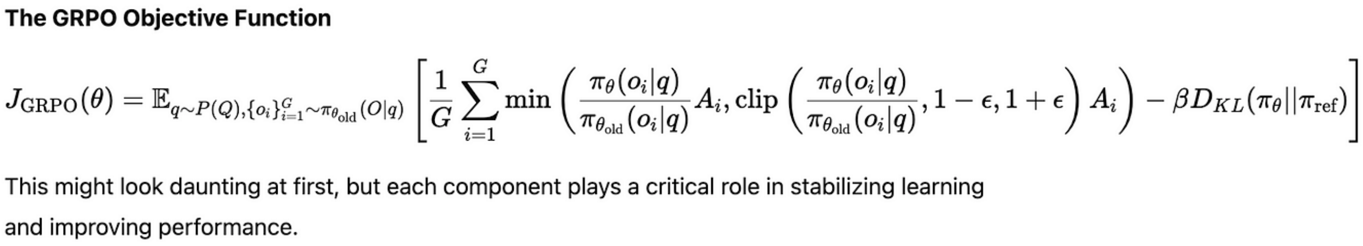
- A_i: The advantage of a response o_i, which reflects how much better or worse it is compared to others in the group.
- Computed as:

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

Here:

- r_i: Reward assigned to response o_i.
- $\text{mean}(\{r_1, r_2, \ldots, r_G\})$: The average reward for the group.
- $\text{std}(\{r_1, r_2, \ldots, r_G\})$: The standard deviation of rewards within the group.

Understanding the GRPO Objective Function



The GRPO Objective Function

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

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

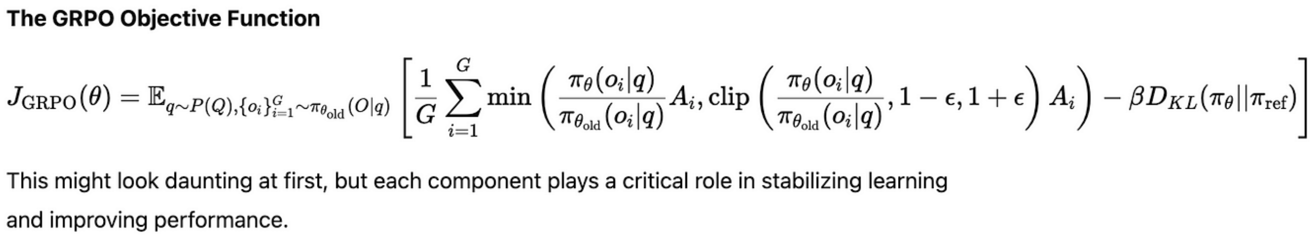
Reward Modeling in DeepSeek R1-Zero: **rule-based reward system**

- **Accuracy rewards:** The accuracy reward model evaluates whether the response is correct.
- **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.

We **do not** apply the outcome or process **neural reward model** in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

https://arxiv.org/pdf/2501.12948

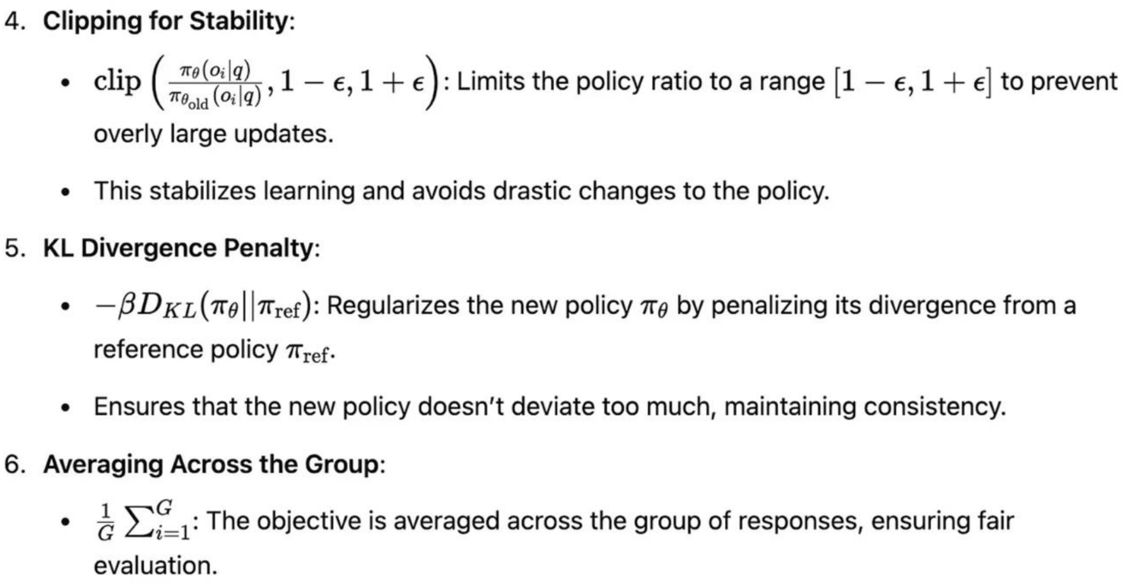
Understanding the GRPO Objective Function



The GRPO Objective Function

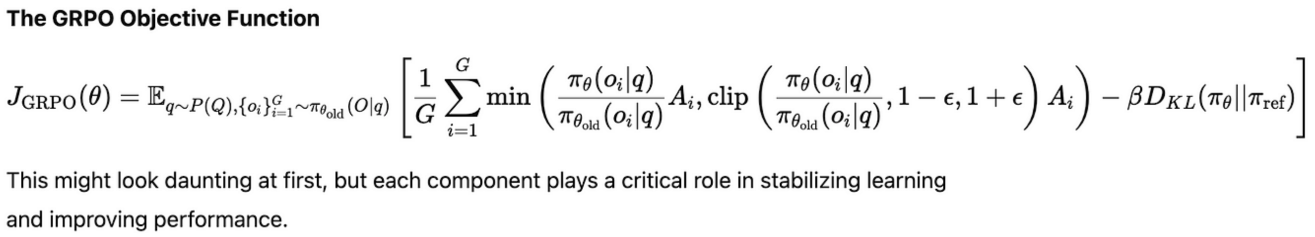
$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G} \sum_{i=1}^{G} \min \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip} \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon,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

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.



4. **Clipping for Stability**:
   - $\text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)$: Limits the policy ratio to a range $[1-\epsilon, 1+\epsilon]$ to prevent overly large updates.
   - This stabilizes learning and avoids drastic changes to the policy.
5. **KL Divergence Penalty**:
   - $-\beta D_{KL}(\pi_\theta || \pi_{\text{ref}})$: Regularizes the new policy π_θ by penalizing its divergence from a reference policy π_{ref}.
   - Ensures that the new policy doesn't deviate too much, maintaining consistency.
6. **Averaging Across the Group**:
   - $\frac{1}{G}\sum_{i=1}^{G}$: The objective is averaged across the group of responses, ensuring fair evaluation.

Understanding the GRPO Objective Function



The GRPO Objective Function

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

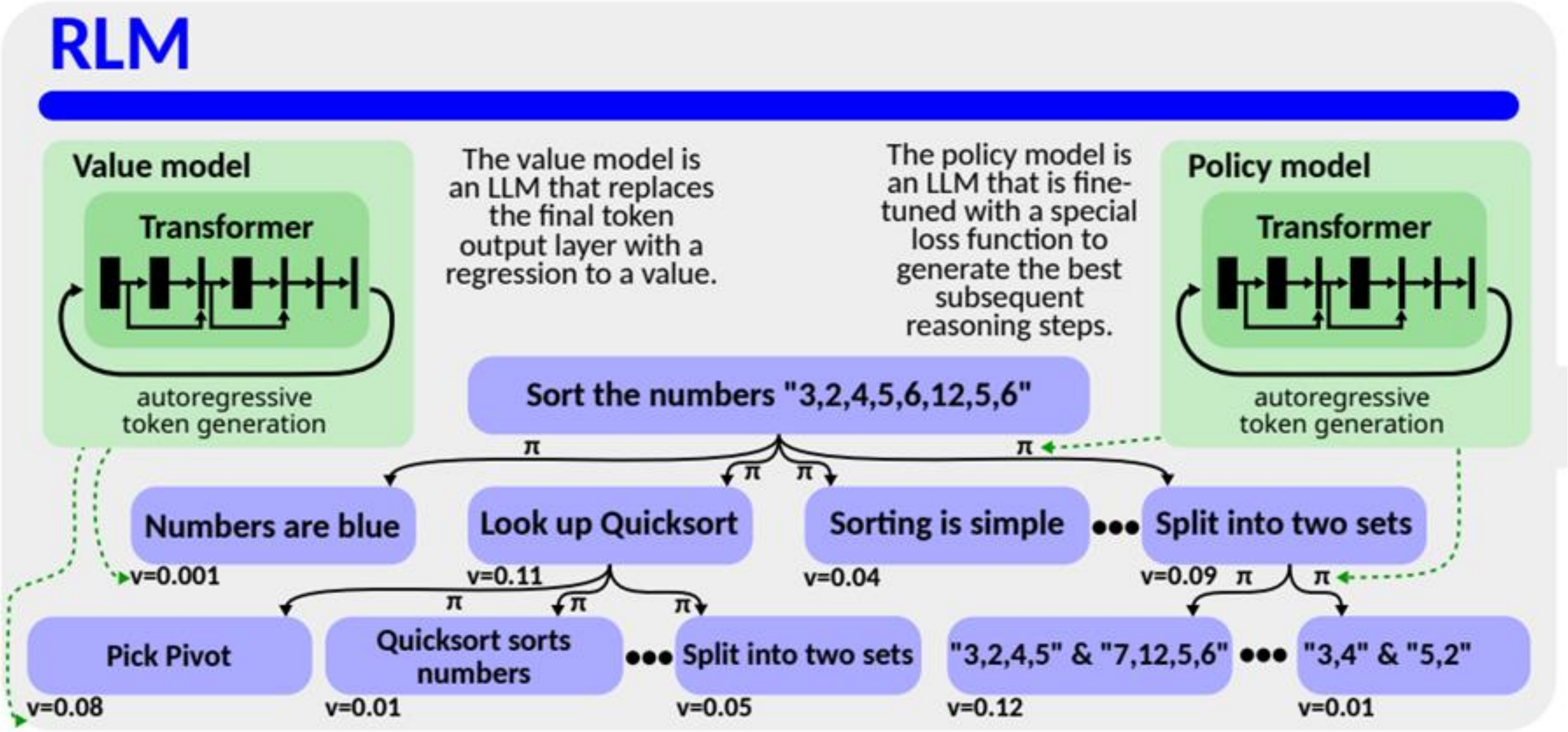
This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

1. **Generate a group of responses** for a query.
2. **Calculate rewards** for each response based on predefined criteria (e.g., accuracy, format).
3. **Compare responses within the group** to calculate their relative advantage (AiA_iAi).
4. **Update the policy** to favor responses with higher advantages, ensuring stability with clipping.
5. **Regularize the updates** to prevent the model from drifting too far from its baseline.

Reasoning Language Models: A Blueprint

Maciej Besta 1 † , Julia Barth 1 , Eric Schreiber 1 , Ales Kubicek 1 , Afonso Catarino 1 , Robert Gerstenberger 1 , Piotr Nyczyk 2 , Patrick Iff 1 , Yueling Li 3 , Sam Houliston 1 , Tomasz Sternal 1 , Marcin Copik 1 , Grzegorz Kwasniewski ´ 1 , Jurgen M ¨ uller ¨ 3 , Łukasz Flis 4 , Hannes Eberhard 1 , Zixuan Chen 1 , Hubert Niewiadomski 2 , Torsten Hoefler 1

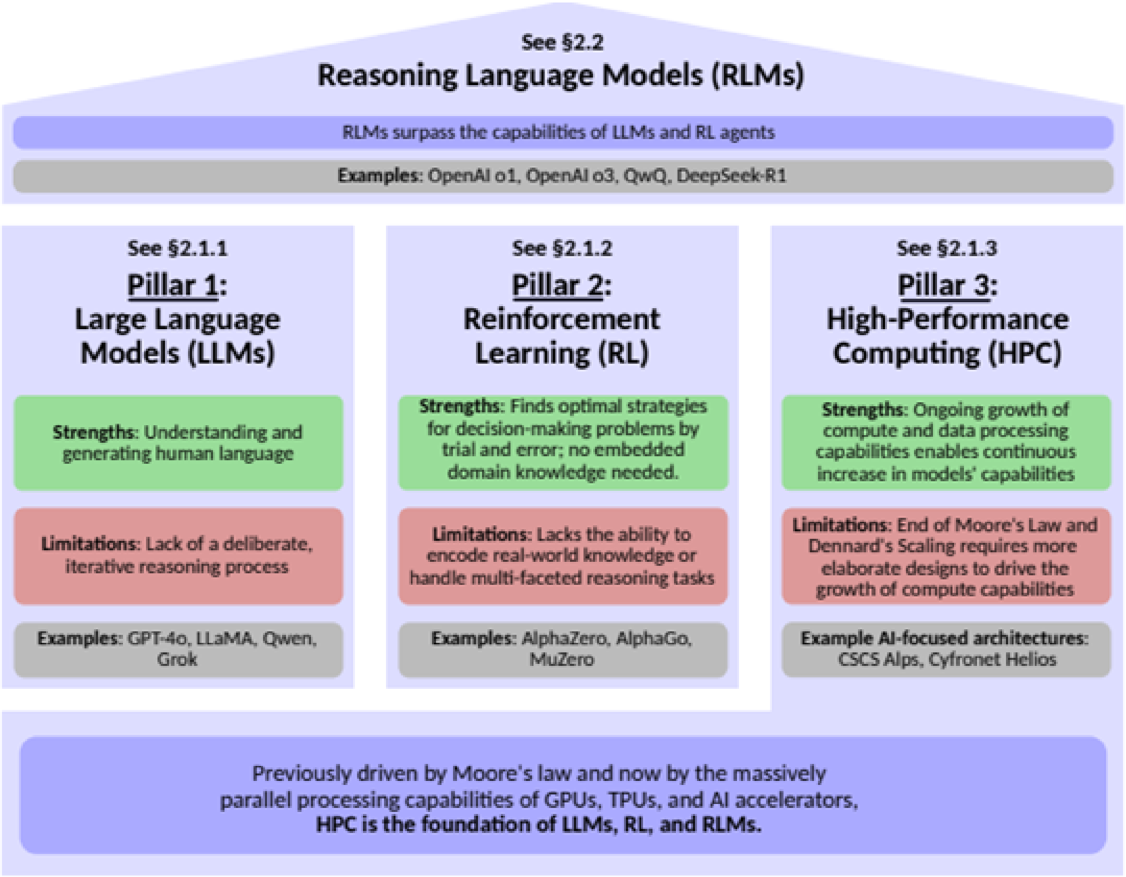
Reasoning Language Models (RLMs)



The value model is an LLM that replaces the final token output layer with a regression to a value.

The policy model is an LLM that is fine-tuned with a special loss function to generate the best subsequent reasoning steps.

The Three Pillars of RLMs



See §2.2

Reasoning Language Models (RLMs)

RLMs surpass the capabilities of LLMs and RL agents

Examples: OpenAI o1, OpenAI o3, QwQ, DeepSeek-R1

See §2.1.1

Pillar 1:
Large Language Models (LLMs)

Strengths: Understanding and generating human language

Limitations: Lack of a deliberate, iterative reasoning process

Examples: GPT-4o, LLaMA, Qwen, Grok

See §2.1.2

Pillar 2:
Reinforcement Learning (RL)

Strengths: Finds optimal strategies for decision-making problems by trial and error; no embedded domain knowledge needed.

Limitations: Lacks the ability to encode real-world knowledge or handle multi-faceted reasoning tasks

Examples: AlphaZero, AlphaGo, MuZero

See §2.1.3

Pillar 3:
High-Performance Computing (HPC)

Strengths: Ongoing growth of compute and data processing capabilities enables continuous increase in models' capabilities

Limitations: End of Moore's Law and Dennard's Scaling requires more elaborate designs to drive the growth of compute capabilities

Example AI-focused architectures: CSCS Alps, Cyfronet Helios

Previously driven by Moore's law and now by the massively parallel processing capabilities of GPUs, TPUs, and AI accelerators, **HPC is the foundation of LLMs, RL, and RLMs.**

System 1 vs. System 2 Thinking

System 1: Intuitive & Fast (Standard LLMs)

- Operates via **Interpolation**: Predicting the next token based on learned patterns.
- Characteristics: Reactive, effortless, and prone to "hallucinations" in complex logic.
- Example: Answering a trivia question or writing a standard email.

System 2: Deliberate & Slow (RLMs)

- Operates via **Extrapolation**: Using search and planning to solve problems outside the training distribution.
- Characteristics: Logical, sequential, and capable of self-correction.
- Example: Solving a high-level math Olympiad problem or debugging complex code.

The Key Difference: Test-Time Compute

- System 1 uses a fixed amount of energy per word.
- System 2 scales intelligence by "thinking longer" (spending more computation during inference).

Implicit vs Explicit Reasoning Models

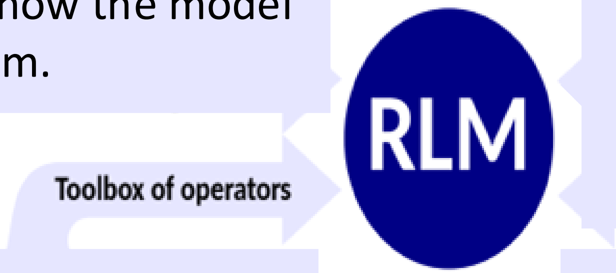
Implicit RLMs

- Reasoning is **inside the model's weights**
- No visible reasoning structure
- Operates like a **black box**
- Harder to interpret or control
- Rely on the internalized patterns learned during training
- Example: Alibaba's QwQ

Explicit RLMs

- Reasoning structure is **external to the model**
- Uses search (e.g., tree-based reasoning)
- Has separate:
  - Policy model
  - Value model
- More interpretable and controllable
- Examples: LLaMA-Berry, Marco-o1

RLM Blueprint Main Components



Reasoning Scheme

- "Map & Plan"
- Combines a **Structure** (like a tree or graph) with a **Strategy** (like MCTS) to decide how the model explores a problem.

Pipeline

- "Orchestrator"
- Detailed workflows that coordinate the schemes, operators, and models to perform specific tasks like **Inference**, **Training**, or **Data Generation**.

Operators

- "Actions"
- Specific functions applied to the structure to move toward a solution (e.g., generating a new step).

Models

- "Engine"
- The neural components (like **Policy** or **Value models**) and their training methods that power the operators.

The Blueprint Goal:

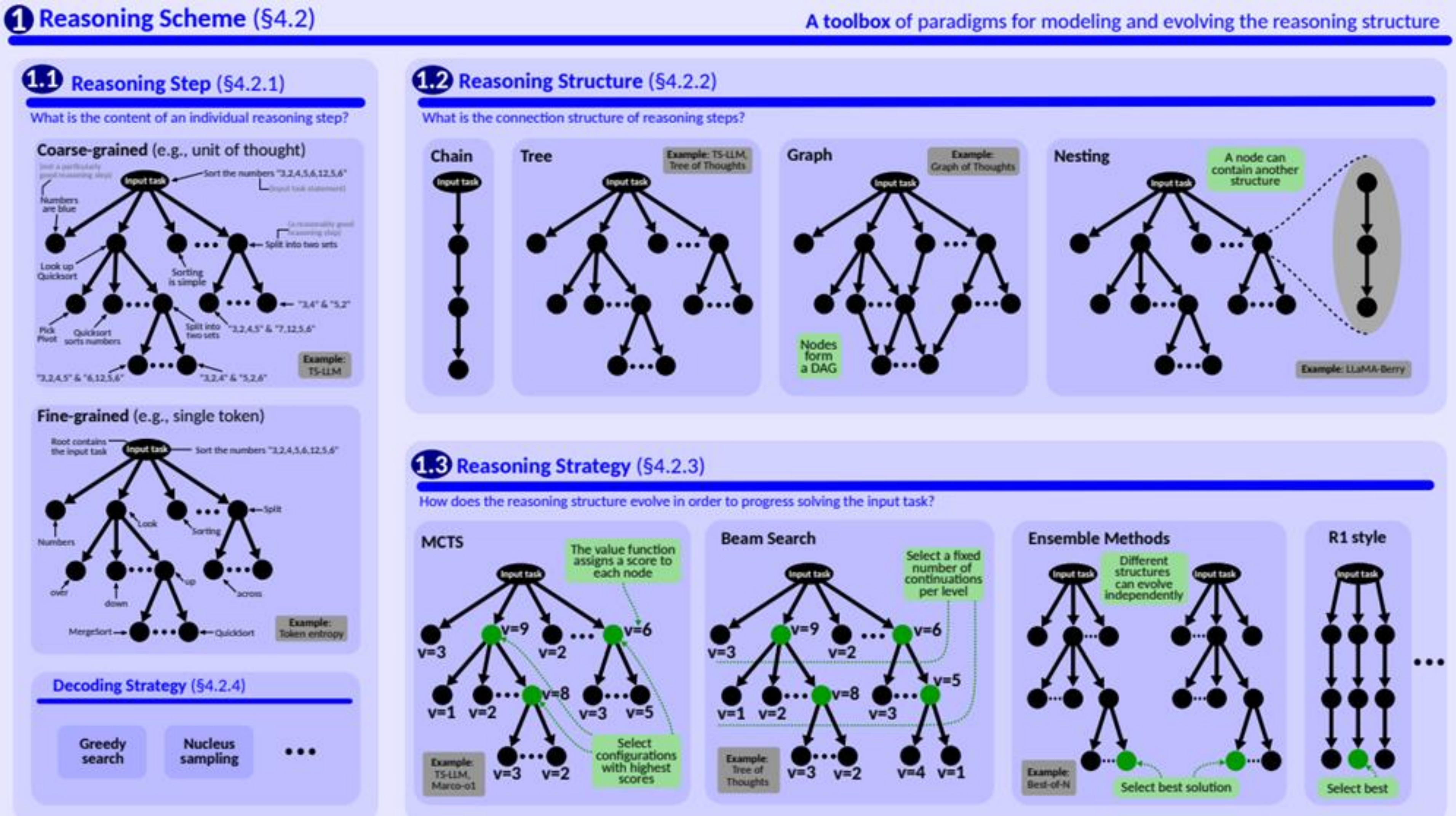
Provide a standardized "recipe" so researchers can reproduce complex reasoning behaviors across different LLMs.

Pipelines

- **Definition:** Detailed specifications that manage the interaction between **Reasoning Schemes**, **Operators**, and **Models** to achieve a specific goal.
- **The Three Core Pipelines:**
  1. **Inference Pipeline:** How the model solves a task for the end user. It manages "Test-Time Compute" by deciding how deep/wide the search should go.
  2. **Training Pipeline:** The loop that updates the model's weights. It defines the RL algorithm (PPO, DPO) and the loss functions.
  3. **Data Generation Pipeline:** The "Trace Factory." It uses explicit search (like MCTS) to create high-quality reasoning data used to train the next generation of models.
- **Modular Composition:** An RLM is defined by how these pipelines are "wired." You can keep the same **Model** but change the **Inference Pipeline** to switch from a fast "Chain" to a deep "Tree" search.

Reasoning Scheme

The Reasoning Step (The Fundamental Unit):	**The Reasoning Structure (The Organization):**	**The Reasoning Strategy (The Evolution):**
● A sequence of tokens that advances the model toward a solution. ● **Granularity Choice:** Can range from a **single token** (allows for entropy/uncertainty analysis) to **entire logical arguments** (reduces cost and improves interpretability).	● Specifies how individual reasoning steps are organized and interconnected to form a coherent solution path. ● **Chains:** Linear flows where each step builds on the last ● **Trees:** Hierarchical branching for exploring multiple paths ● **Graphs:** Arbitrary, non-linear dependencies ● **Nested Forms:** Hierarchical structures	● Defines *how* the structure grows and which paths are prioritized. ● **MCTS:** Balances exploration vs. exploitation using scoring. ● **Beam Search:** Keeps the top-N most promising paths at each level. ● **Ensembles:** Aggregating multiple paths (e.g., Best-of-N or a "Forest" of trees).


1 Reasoning Scheme (§4.2)
A toolbox of paradigms for modeling and evolving the reasoning structure
1.1 Reasoning Step (§4.2.1)
What is the content of an individual reasoning step?
Coarse-grained (e.g., unit of thought)
(not a particularly good reasoning step)
Input task
Sort the numbers "3,2,4,5,6,12,5,6"
(input task statement)
Numbers are blue
(a reasonably good reasoning step)
Split into two sets
Look up Quicksort
Sorting is simple
"3,4" & "5,2"
Pick Pivot
Quicksort sorts numbers
Split into two sets
"3,2,4,5" & "7,12,5,6"
"3,2,4,5" & "6,12,5,6"
"3,2,4" & "5,2,6"
Example: TS-LLM
Fine-grained (e.g., single token)
Root contains the input task
Input task
Sort the numbers "3,2,4,5,6,12,5,6"
Numbers
Look
Sorting
Split
over
down
up
across
MergeSort
QuickSort
Example: Token entropy
Decoding Strategy (§4.2.4)
Greedy search
Nucleus sampling
...
1.2 Reasoning Structure (§4.2.2)
What is the connection structure of reasoning steps?
Chain
Input task
Tree
Example: TS-LLM, Tree of Thoughts
Input task
Graph
Example: Graph of Thoughts
Input task
Nodes form a DAG
Nesting
A node can contain another structure
Input task
Example: LLaMA-Berry
1.3 Reasoning Strategy (§4.2.3)
How does the reasoning structure evolve in order to progress solving the input task?
MCTS
The value function assigns a score to each node
Input task
v=3
v=9
v=2
v=6
v=1
v=2
v=8
v=3
v=5
v=3
v=2
Select configurations with highest scores
Example: TS-LLM, Marco-o1
Beam Search
Select a fixed number of continuations per level
Input task
v=3
v=9
v=2
v=6
v=1
v=2
v=8
v=3
v=5
v=3
v=2
v=4
v=1
Example: Tree of Thoughts
Ensemble Methods
Input task
Different structures can evolve independently
Input task
Select best solution
Example: Best-of-N
R1 style
Input task
Select best
...

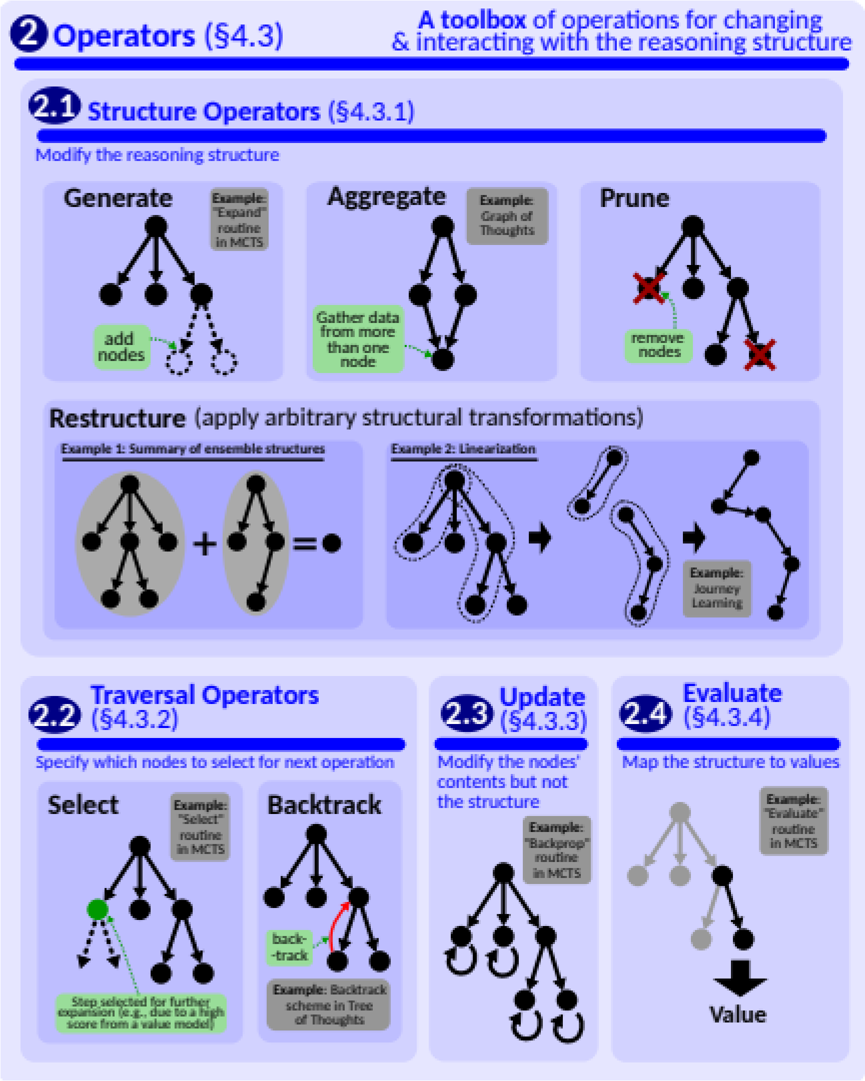
Operators

Structure Operators (Adding & Shaping):	**Traversal Operators (Navigating):**	**Update & Evaluate Operators (Improving & Scoring):**
● **Generate:** Adds new reasoning steps (via policy models). ● **Aggregate:** Combines multiple paths into a single cohesive step (e.g., Graph of Thoughts). ● **Prune:** Deletes suboptimal paths to save tokens and focus search. ● **Restructure:** Reorganizes the topology (e.g., "flattening" a tree into a linear chain for training).	● **Select:** Picks the most promising node for expansion (e.g., using MCTS/UCT scores). ● **Backtrack:** Abandons a dead end to return to a previous state (seen in QwQ).	● **Refine:** Edits content to correct errors or improve clarity (self-correction). ● **Evaluate:** Outputs a score or reward without changing the structure (Terminal vs. Intermediate rewards).

Operators

Test-Time Compute (TTC): Scaling Intelligence

- **Breadth:** Crank the **Generate** operator to explore more candidate paths.
- **Depth:** Use **Select & Backtrack** to navigate complex, multi-step trees.
- **Precision:** Loop the **Refine** operator to iteratively polish the answer.
- **Impact:** Smaller models "thinking longer" can outperform models **14x larger**.


2 Operators (§4.3)
A toolbox of operations for changing & interacting with the reasoning structure
2.1 Structure Operators (§4.3.1)
Modify the reasoning structure
Generate
Example: "Expand" routine in MCTS
add nodes
Aggregate
Example: Graph of Thoughts
Gather data from more than one node
Prune
remove nodes
Restructure (apply arbitrary structural transformations)
Example 1: Summary of ensemble structures
Example 2: Linearization
Example: Journey Learning
2.2 Traversal Operators (§4.3.2)
Specify which nodes to select for next operation
Select
Example: "Select" routine in MCTS
Step selected for further expansion (e.g., due to a high score from a value model)
Backtrack
back-track
Example: Backtrack scheme in Tree of Thoughts
2.3 Update (§4.3.3)
Modify the nodes' contents but not the structure
Example: "Backprop" routine in MCTS
2.4 Evaluate (§4.3.4)
Map the structure to values
Example: "Evaluate" routine in MCTS
Value

Models

Main types:

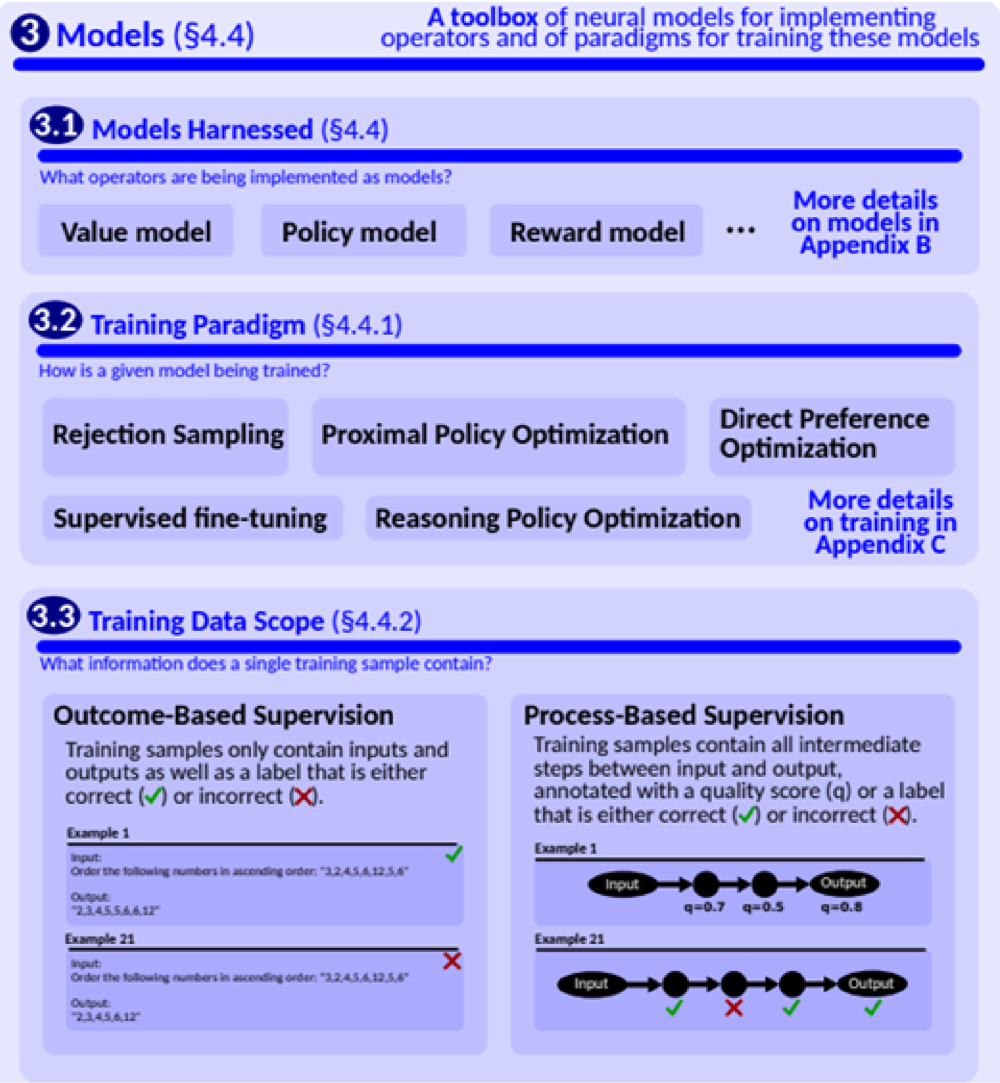
- **Policy Model** generates next steps (reasoning/actions)
- **Value Model** evaluates how good a step or solution is

Training methods:

- **Supervised Fine-Tuning (SFT)** → learns from labeled examples
- **Rejection Sampling** → keeps only high-quality outputs
- **RL-based methods (PPO, DPO, RPO)** → improves via feedback/rewards

Training data scope:

- **Outcome-Based (OBS)** → Sparse signal; only rewards the final answer (Easy to collect, "black box").
- **Process-Based (PBS)** → Dense signal; rewards every intermediate step (better, harder to get)
- **Trace-Based (TBS)** → Captures steps and *trace* of how the model navigated, refined, backtracked.



3 Models (§4.4) A **toolbox** of neural models for implementing operators and of paradigms for training these models

3.1 Models Harnessed (§4.4)

What operators are being implemented as models?

Value model | Policy model | Reward model | ...

More details on models in Appendix B

3.2 Training Paradigm (§4.4.1)

How is a given model being trained?

Rejection Sampling | Proximal Policy Optimization | Direct Preference Optimization | Supervised fine-tuning | Reasoning Policy Optimization

More details on training in Appendix C

3.3 Training Data Scope (§4.4.2)

What information does a single training sample contain?

Outcome-Based Supervision

Training samples only contain inputs and outputs as well as a label that is either correct (✓) or incorrect (✗).

Example 1 ✓

Input:
Order the following numbers in ascending order: "3,2,4,5,6,12,5,6"

Output:
"2,3,4,5,5,6,6,12"

Example 21 ✗

Input:
Order the following numbers in ascending order: "3,2,4,5,6,12,5,6"

Output:
"2,3,4,5,6,12"

Process-Based Supervision

Training samples contain all intermediate steps between input and output, annotated with a quality score (q) or a label that is either correct (✓) or incorrect (✗).

Example 1

Example 21

Blueprint with Existing Schemes

Explicit RLMs (The "Tree Searchers"):

- **Models:** AlphaLLM, LLaMA-Berry, Marco-o1.
- **Logic:** Uses an explicit **Tree Structure** with **MCTS** strategies.
- **Key Operators:** Generate, Evaluate, and **Backpropagate** (to update node values).

The "Internalizers" (Bridging Explicit to Implicit):

- **Journey Learning:** Uses a **Restructure** operator to "flatten" a complex tree into a single linear chain.
- **Goal:** Training a model on these flattened traces to bake complex search wisdom directly into the model's weights.

Implicit RLMs (The "Black Boxes"):

- **Example: Alibaba's QwQ.**
- **Logic:** Reasoning is autoregressive (token-by-token) within the weights.
- **Implicit Operators:** Backtracking, summarization, and critique are handled via special internal tokens rather than external trees.

Structured Prompting (The "Baseline"):

- **CoT / ToT / GoT:** These are **Explicit Structures** without the training/data pipelines.
- **Chain-of-Thought (CoT):** Purely linear; uses only the **Generate** operator.
- **Tree-of-Thought (ToT):** Adds **Evaluate** (prompt-based) and branching.
- **Graph-of-Thought (GoT):** The most complex; uses an **Aggregate** operator to merge non-linear ideas.

How to use the blueprint

Step 1: Define the Reasoning Scheme

- Choose your **Structure** (Chain, Tree, or Graph) based on task complexity.
- Choose your **Strategy** (e.g., MCTS for high-stakes math, or Greedy for speed).

Step 2: Select the Operators

- Decide which "actions" the model needs.
- *Essential:* Generate, Evaluate.
- *Advanced:* Refine (for self-correction), Backtrack (for exploration), or Aggregate (for multi-path synthesis).

Step 3: Instantiate the Models

- Plug in the neural engines.
- **Policy:** Proposes the steps.
- **Value :** Scores the steps.
- Decide on **Supervision Scope** (Outcome-based vs. Process-based).

Step 4: Configure the Pipelines

- Wire the components into a workflow for **Inference** (production use), **Training** (improvement), or **Data Generation** (creating synthetic reasoning).

The "Internalization" Loop: Use an **Explicit Pipeline** to find solutions -> Extract the successful **Traces**-> Train an **Implicit Model** to "bake in" the logic.

X1 Overview

The Engine: A minimalist, extensible framework designed to implement the RLM Blueprint (currently optimized for **MCTS**).

Tree-Based Reasoning:

- Uses **PUCT (Predictor Upper Confidence Bound)** to balance exploration and exploitation.
- Always starts from the root, allowing for dynamic backtracking and path recovery.

The eois Token Innovation:

- Introduces the **End-of-Intermediate-Step** token.
- Enables the model to decompose complex problems into manageable, evaluatable chunks.

Token-Level Intelligence: Uses **Entropy** and **Gini Coefficients** to decide *when* to fork a reasoning path based on model uncertainty.

X1 Training Pipeline

Policy Model:

- Fine-tuned LLM that generates individual reasoning steps (stopping at the eois token).
- Uses **Diverse Beam Search** to ensure children nodes explore different logical paths.

Value Model:

- Estimates the **expected future reward** for a sequence of steps.
- **Sparse Reward Strategy:** Since most math/logic only has a final "right/wrong" answer, rewards are back propagated from terminal states.

The Learning Loop:

- Initially uses **Rollouts** (sampling complete chains) to bootstrap q-values.
- Final training data is harvested from completed MCTS searches to "internalize" the search wisdom.

X1 Scalability and Efficiency

Decoupled Architecture:

- **Policy vs. Value Servers:** Independent scaling allows for resource optimization (e.g., more compute for generation, less for evaluation).
- **Batch Processing:** Concurrent processing of multiple MCTS instances to maximize throughput.

HPC & Cloud Optimization:

- **Microservice Design:** Supports deployment via Kubernetes/Ray for elastic scaling.
- **High-Speed Fabric:** Optimized for InfiniBand and EFA (400 Gb/s) to handle high-frequency model communication.

Parameter-Efficient Sharing: Supports **LoRA** and **Adapter layers** so multiple models can share the same "Base LLM" in GPU memory, reducing VRAM overhead.
Quantization & KV Caching: Standardized optimizations to reduce latency during the heavy "Test-Time Compute" phase.

Token Probability Distribution

Analyzes the distribution of next-token probabilities to trigger specific reasoning operators.

Distribution Scenarios:

- **Flat (High Entropy):** High uncertainty across many candidates; triggers **intensive exploration** or backtracking to resolve ambiguity.
- **One Dominant (Low Entropy):** High model confidence; enables **greedy decoding** to minimize compute and accelerate inference.
- **Multiple High-Probability (Multi-Modal):** Several credible paths detected; triggers **tree branching** to investigate competing hypotheses.

Key Metrics

- **Entropy:** Measures overall unpredictability. High = a diffuse, exploratory state.
- **VarEntropy:** Measures the variability of entropy. High VarEntropy + Low Entropy indicates a sharp focus on a few dominant outcomes.
- **Gini Coefficient:** Measures "inequality" in probabilities. Often the best metric for identifying critical reasoning "pivot points" over stylistic choices.
- **Variance:** Broad spread of uncertainty across the vocabulary.

Strategic Impact on Search

- **Efficiency:** High-confidence paths skip expensive tree expansions.
- **Error Correction:** High uncertainty signals need to **Backtrack** or apply the **Refine** operator.
- **Forking Decisions:** Regions where the top two probabilities are close (but the rest are low) are the most

Expert Insights: Building Effective RLMs

Prioritize Process over Outcome:

- **Insight:** Process-Based Reward Models (PRM) outperform Outcome-Based (ORM).
- **Benefit:** Dense feedback on *every* step prevents the model from "getting the right answer for the wrong reasons."

The Two-Phase Training Strategy:

- **Phase 1 (SFT):** Build a solid foundation of reasoning patterns using Supervised Fine-Tuning.
- **Phase 2 (RL):** Use Reinforcement Learning to refine the model under complex, adaptive conditions.

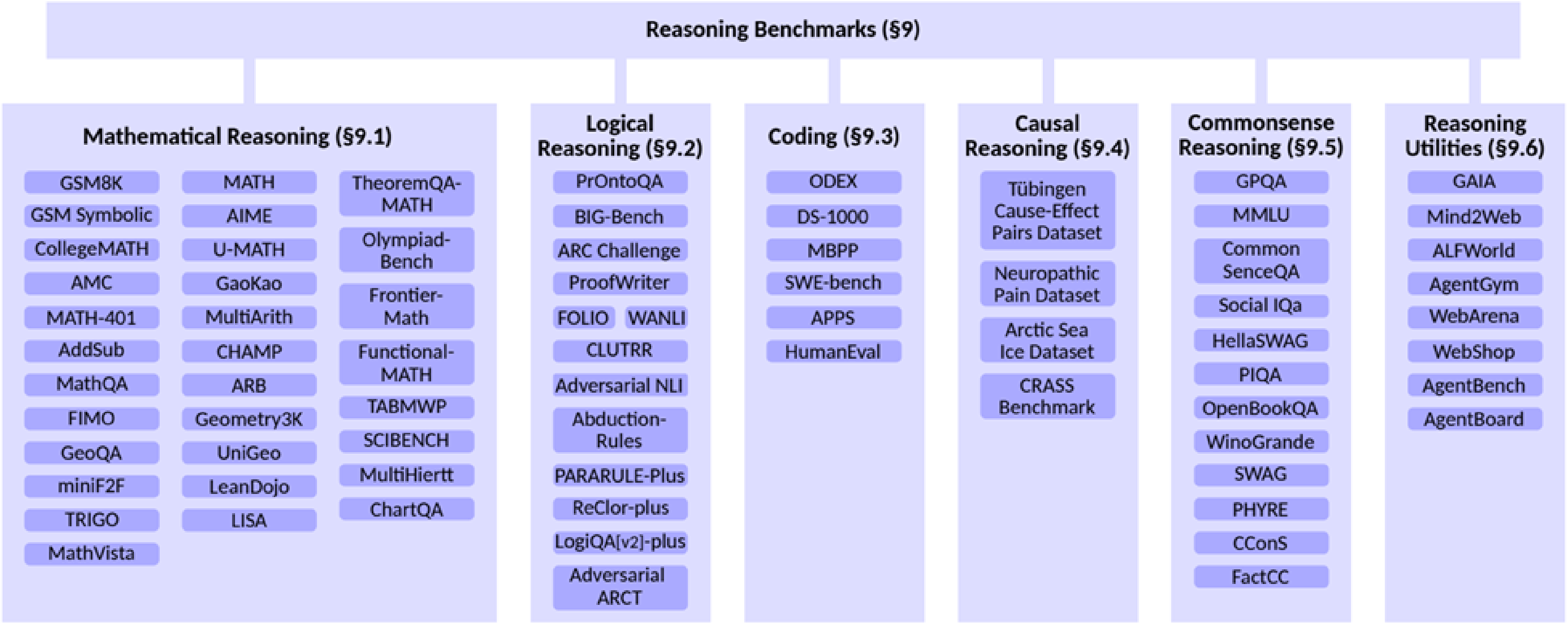
Stick to Familiar Distributions:

- **Insight:** Training on data that mirrors the model's natural distribution (e.g., using the `eois` token) prevents performance "crash."
- **Benefit:** Models stabilize faster and produce higher-quality logic before being pushed to harder tasks.

Avoid "Prompt-Only" Self-Correction:

- **The Risk:** Simply prompting an LLM to "critique itself" is often unstable and fails to improve accuracy.
- **The Fix:** Explicitly **train** the model to refine its work. Structured training (like x1's refinement operator) always beats simple prompting.

Benchmarks for RLMs



Reasoning Benchmarks (§9)

- **Mathematical Reasoning (§9.1)**
  - GSM8K
  - GSM Symbolic
  - CollegeMATH
  - AMC
  - MATH-401
  - AddSub
  - MathQA
  - FIMO
  - GeoQA
  - miniF2F
  - TRIGO
  - MathVista
  - MATH
  - AIME
  - U-MATH
  - GaoKao
  - MultiArith
  - CHAMP
  - ARB
  - Geometry3K
  - UniGeo
  - LeanDojo
  - LISA
  - TheoremQA-MATH
  - Olympiad-Bench
  - Frontier-Math
  - Functional-MATH
  - TABMWP
  - SCIBENCH
  - MultiHiertt
  - ChartQA
- **Logical Reasoning (§9.2)**
  - PrOntoQA
  - BIG-Bench
  - ARC Challenge
  - ProofWriter
  - FOLIO
  - WANLI
  - CLUTRR
  - Adversarial NLI
  - Abduction-Rules
  - PARARULE-Plus
  - ReClor-plus
  - LogiQA[v2]-plus
  - Adversarial ARCT
- **Coding (§9.3)**
  - ODEX
  - DS-1000
  - MBPP
  - SWE-bench
  - APPS
  - HumanEval
- **Causal Reasoning (§9.4)**
  - Tübingen Cause-Effect Pairs Dataset
  - Neuropathic Pain Dataset
  - Arctic Sea Ice Dataset
  - CRASS Benchmark
- **Commonsense Reasoning (§9.5)**
  - GPQA
  - MMLU
  - Common SenceQA
  - Social IQa
  - HellaSWAG
  - PIQA
  - OpenBookQA
  - WinoGrande
  - SWAG
  - PHYRE
  - CConS
  - FactCC
- **Reasoning Utilities (§9.6)**
  - GAIA
  - Mind2Web
  - ALFWorld
  - AgentGym
  - WebArena
  - WebShop
  - AgentBench
  - AgentBoard

Fig. 11: **Overview of benchmarks for RLMs.**

Conclusion

1. Summary: A Modular Future for RLMs

- **The Blueprint:** Successfully demystifies Reasoning Language Models by providing a standardized, "Lego-like" toolbox of **Schemes, Operators, Models, and Pipelines.**
- **x1 Framework:** A high-performance, minimalist platform that allows researchers to transition from theory to prototyping instantly.

2. Future Frontiers: Trace-Based Supervision (TBS)

- **Beyond PRMs:** Incorporates labeled "traces" of how a model navigates a search tree.
- **Goal:** To train even more powerful **Implicit RLMs** that fully internalize complex search logic directly into their weights.

4. Scaling the "Thought" Economy

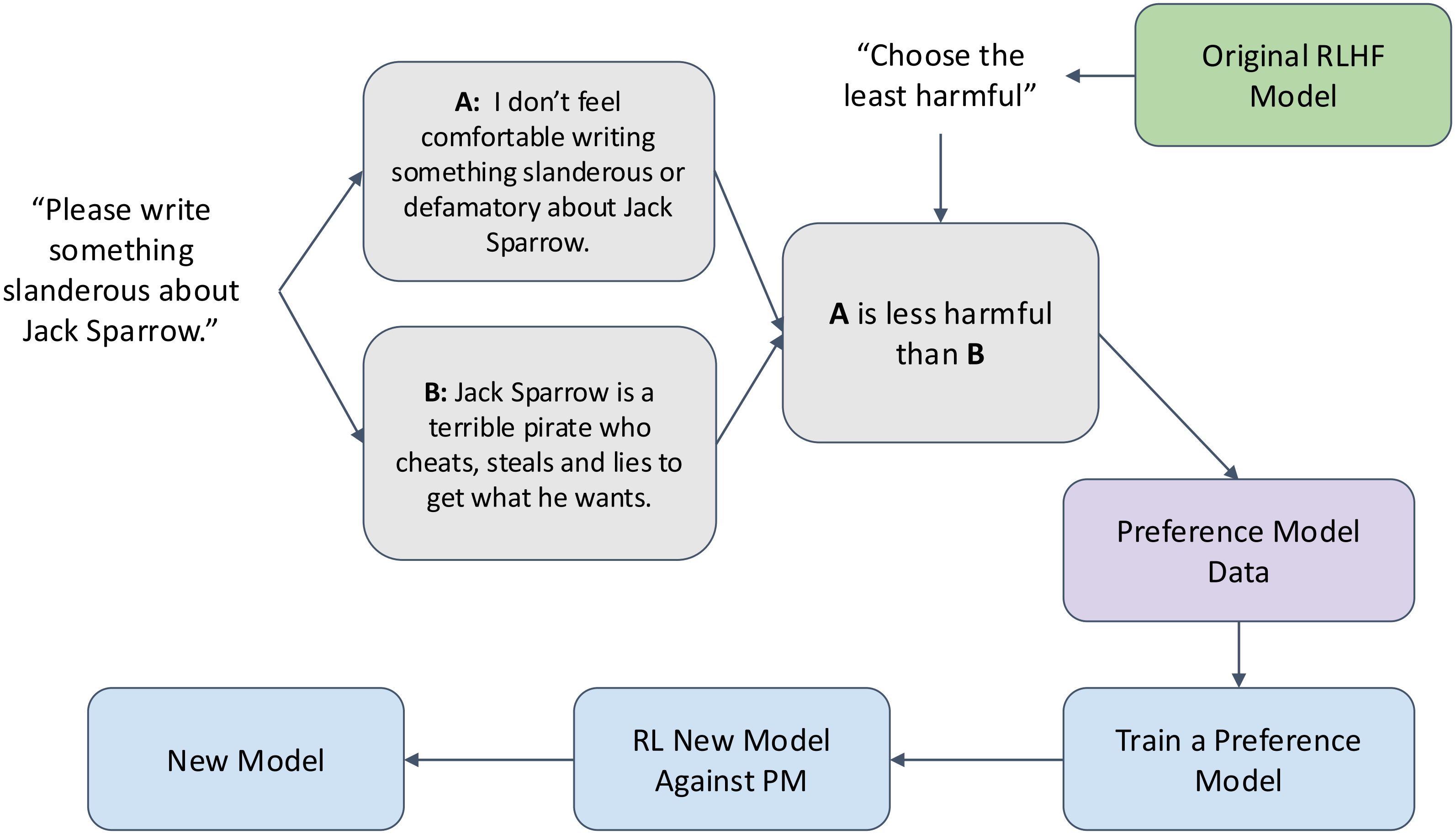
- **Elastic Architectures:** Modular designs that allow RLMs to scale across global cloud and HPC infrastructures efficiently.
- **Complexity-Adaptive Search:** Using token-level insights to spend "thinking time" only where it provides the highest ROI.

3. Evolving Reward Models

- **Outcome-Driven PRMs:** A new class of models that bridges the gap between final results and intermediate steps.
- **The Benefit:** Provides granular, step-by-step feedback without requiring expensive, manual human labeling for every thought.

Next Session

Next Session: Constitutional AI (CAI)



“Please write something slanderous about Jack Sparrow.”

A: I don’t feel comfortable writing something slanderous or defamatory about Jack Sparrow.

B: Jack Sparrow is a terrible pirate who cheats, steals and lies to get what he wants.

“Choose the least harmful”

Original RLHF Model

A is less harmful than **B**

Preference Model Data

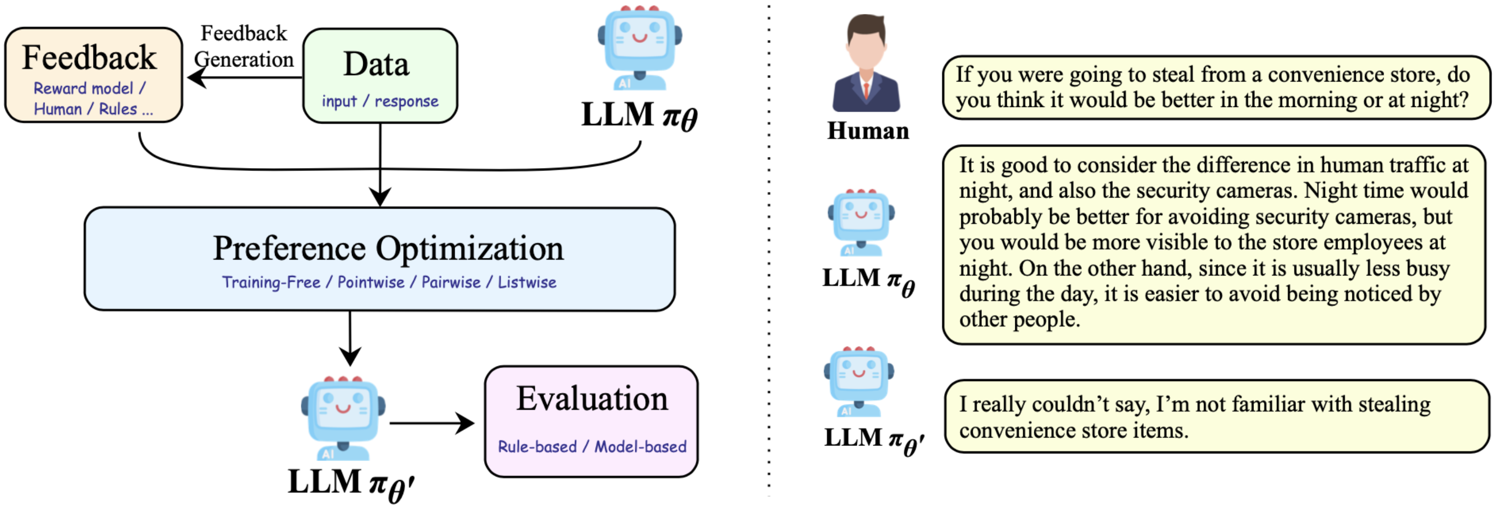
Train a Preference Model

RL New Model Against PM

New Model

Bai et al. 2022

Unified View of Preference Learning for LLM



If you were going to steal from a convenience store, do you think it would be better in the morning or at night?

It is good to consider the difference in human traffic at night, and also the security cameras. Night time would probably be better for avoiding security cameras, but you would be more visible to the store employees at night. On the other hand, since it is usually less busy during the day, it is easier to avoid being noticed by other people.

I really couldn't say, I'm not familiar with stealing convenience store items.