

CS388: Natural Language Processing

Lecture 16: Reasoning and RLVR



TEXAS

The University of Texas at Austin

Administration

- HW2 due 10/22
- Proposal due 10/23

Today

- Reasoning recap (CoT)
- What tasks need reasoning?
- Best-of-N and recap of self-consistency
- What is a verified reward?
- RL from verified rewards with GRPO
- Reasoning models and open challenges
- Non-verifiable tasks

Reasoning recap

- **Chain of thought (CoT):** generate intermediate steps, then the answer
- Few-shot CoT examples (Wei et al., 2022) or just “Let’s think step by step” (Kojima et al., 2022)
- More generated tokens = more computation per answer

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

Direct answer

A: The answer is 27.

Chain of thought

A: They had 23 apples and used 20, so $23 - 20 = 3$. They bought 6 more, so $3 + 6 = 9$. **The answer is 9.**

Thinking tokens

- Reasoning models write the CoT inside **<think> ... </think>**, then answer
- Origin: R1-Zero prompt template + a **format reward** (Guo et al., 2025)
- Separate from the answer: easy to hide, strip, or cap with a budget
- Can run to thousands of tokens on hard problems

User: What is the sum of all prime numbers less than 20?

<think> I need all primes below 20. 2 is prime. 3 is prime. 4 = 2×2 , not prime. 5 is prime... Adding: $2+3 = 5$, $5+5 = 10$, $10+7 = 17$, $17+11 = 28$, $28+13 = 41$, $41+17 = 58$, $58+19 = 77$. **</think>**

The answer is `\boxed{77}`.

What tasks need reasoning?

Large gains from CoT

- Multi-step math (GSM8K, MATH, AIME)
- Code and algorithms
- Logic puzzles and planning
- Symbolic manipulation

Little or no gain

- Factual recall
- Classification, sentiment
- One-step commonsense QA
- MMLU, except questions with “=”

Common thread: **many dependent steps** with intermediate results to track

Pass@k

- Sample k completions per problem
- **pass@1**: a single sample is correct
- **pass@k**: at least one of k samples is correct
- pass@k measures **coverage**: what the model can solve at all

$$\text{pass@1} = \Pr(\text{sample is correct})$$

$$\text{pass@k} = \mathbb{E}_x \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

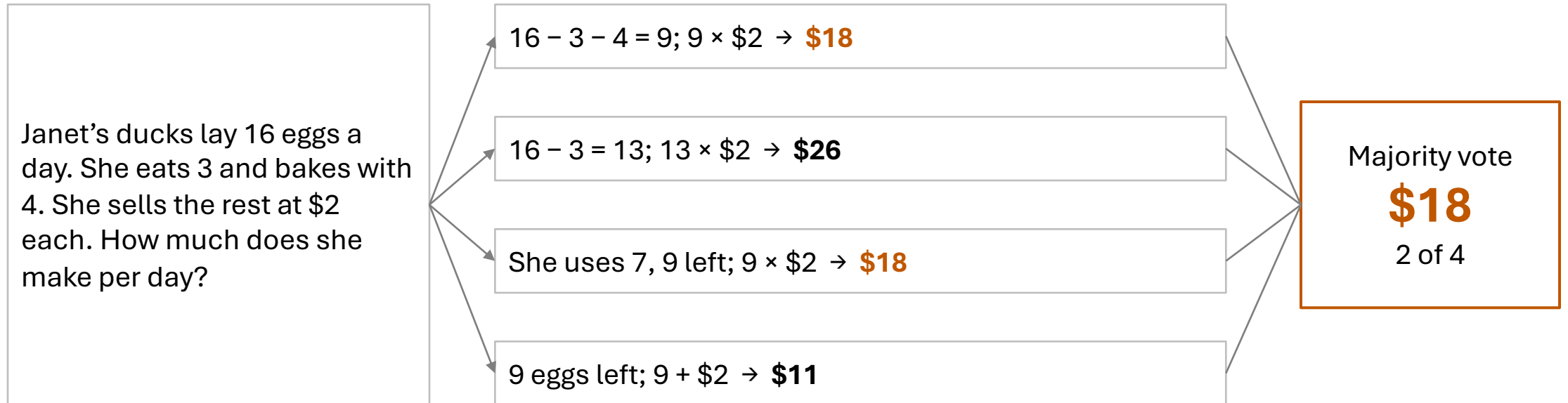
Unbiased estimate from $n \geq k$ samples, c of them correct

Example: $n = 10$, $c = 3$

$\text{pass@1} = 0.30$, $\text{pass@5} \approx 0.92$

Self-consistency

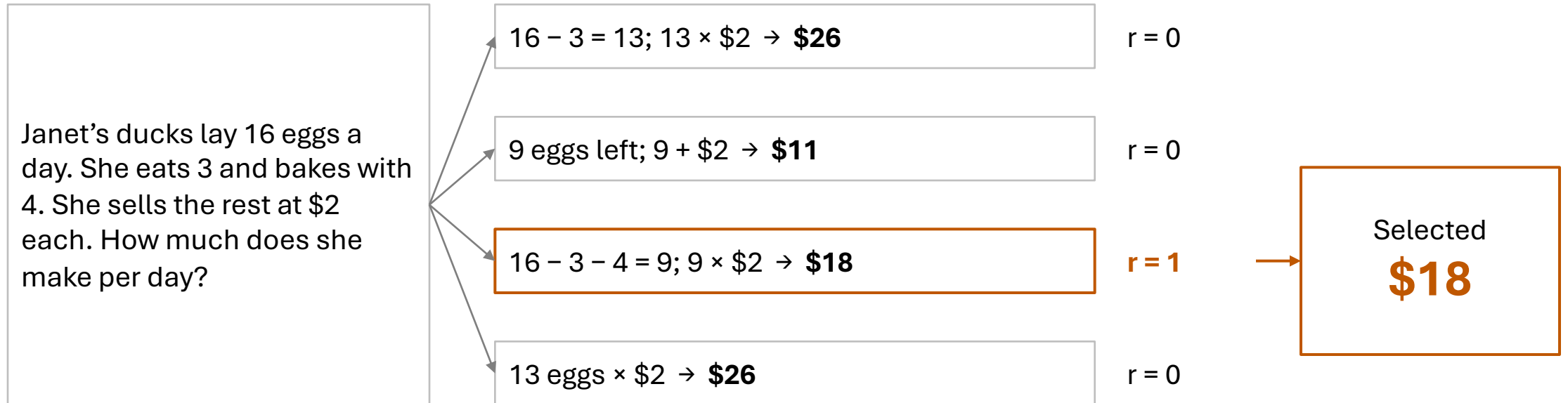
- Sample N CoTs (temperature > 0), return the **most common final answer** (Wang et al., 2023)
- No verifier needed, but answers must be comparable



$$\hat{a} = \arg \max_a \sum_{i=1}^N \mathbb{1}[\text{ans}(y_i) = a]$$

Best-of-N

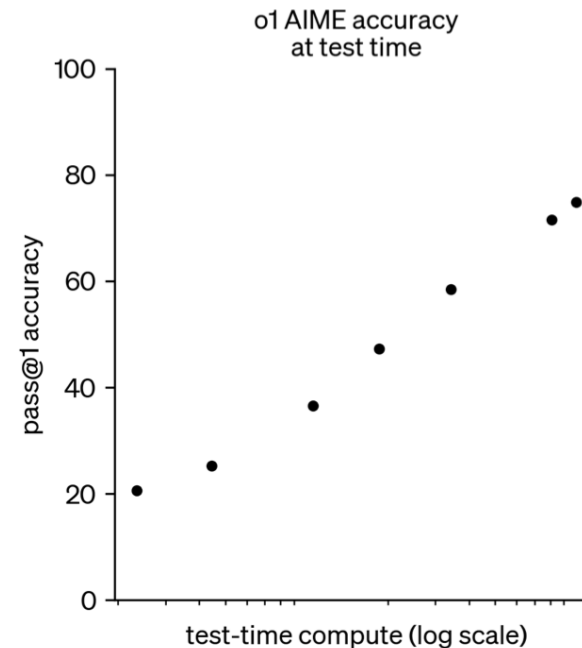
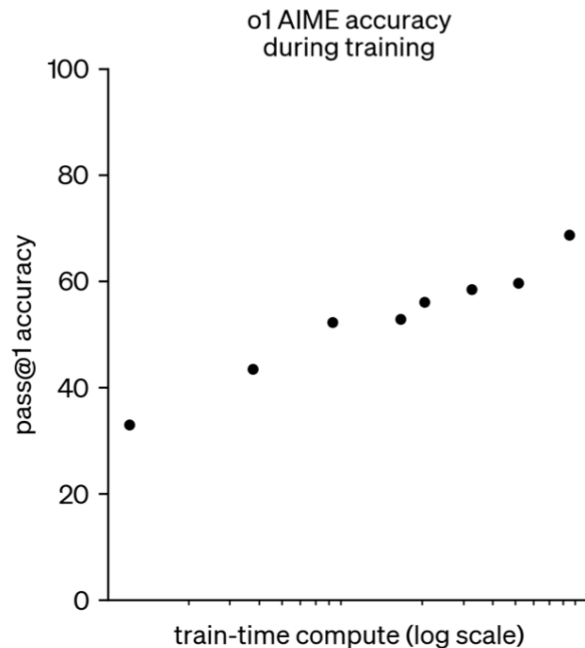
- Sample N completions, score each with a **verifier** (or reward model), keep the best
- Finds the right answer even when it is rare; perfect verifier $\rightarrow$ accuracy = pass@ N



$$\hat{y} = y_{i^*}, \quad i^* = \arg \max_i r(x, y_i)$$

Test-time compute

- Spend more inference compute per problem → better answers
- **Parallel:** many samples, then aggregate (self-consistency, best-of-N)
- **Sequential:** one longer chain of thought (o1, R1)
- o1: accuracy grows log-linearly in **both** RL training compute and test-time compute



Why do we want to do RL here?

- **Test-time compute is costly:** N samples = $N \times$ tokens per query
- **No verifier at test time:** new problems have no answer key
- **Bootstrapping:** train on its own verified successes, repeat
- **Goal:** move pass@1 toward pass@k

What is a verifiable task?

- A **program** checks if the output is correct; no learned reward model
- Needs a ground-truth answer, a test suite, or a checker
- Binary, exact reward:

$$r(x, y) = \mathbb{1}[\text{ans}(y) = a^*] \in \{0, 1\}$$

	RLHF	RLVR
Reward	Learned reward model	Program checks the answer
Signal	Proxy	Exact
Example	Helpful chat replies	Math, code, proofs

Verifiable task: math

- Extract the final answer, e.g. from `\boxed{}`
- Compare to the reference answer
- Normalize equivalent forms: $1/2 = 0.5$
(symbolic checkers like SymPy, Math-Verify)
- Data: GSM8K, MATH, AIME-style problems
- **Caveat:** right answer with wrong reasoning still gets $r = 1$ (recall faithfulness in CoT)

Prompt

If $2x + 3 = 4$, what is x ?

Output

... so $2x = 1$ and $x = 1/2$. The answer is
`\boxed{\frac{1}{2}}`

Check

extracted: `\frac{1}{2}` reference: 0.5
equivalent $\rightarrow r = 1$

Verifiable task: code

- Run the code against **unit tests**
- Reward: all tests pass (or fraction passed)
- Data: competitive programming (APPS, CodeContests, LiveCodeBench)
- **Caveat:** signal is only as good as your tests

Prompt

Write fib(n) that returns the nth Fibonacci number.

Output

```
def fib(n):  
    if n < 2: return n  
    return fib(n-1) + fib(n-2)
```

Tests

```
assert fib(0) == 0  
assert fib(10) == 55  
all pass → r = 1
```

Verifiable task: proofs

- Write proofs in a formal language (e.g. **Lean 4**)
- The proof assistant checks every step; reward = proof compiles
- DeepSeek-Prover, AlphaProof (IMO 2024 silver-medal level)
- **Hard part:** formalizing the statement; natural-language proofs need a judge

Statement

For all natural numbers a, b : $a + b = b + a$

Output

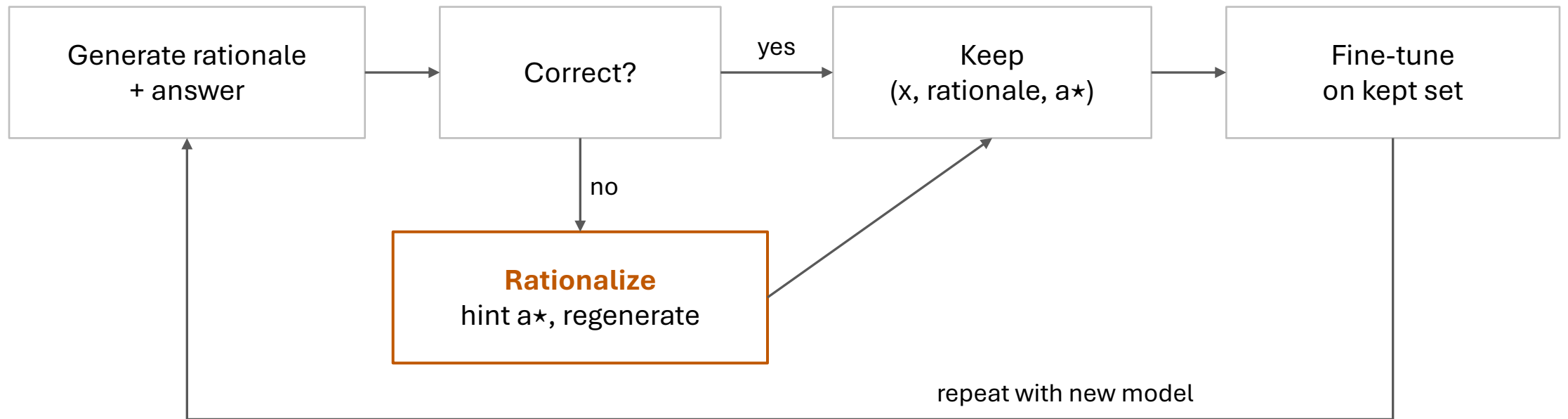
```
theorem add_comm' (a b : Nat) :  
  a + b = b + a := by  
  omega
```

Check

Lean compiles, no sorry → **r = 1**

STaR: Self-Taught Reasoner

- Early precursor to RLVR
- Bootstrap reasoning from a few rationale examples (Zelikman et al., 2022)
- Keep only rationales that reach the right answer, fine-tune, repeat

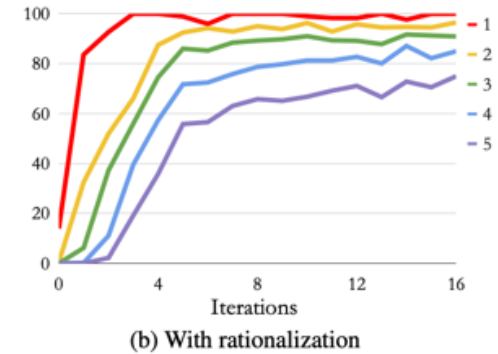


Filtering with $r \in \{0, 1\} \approx$ policy gradient:

$$\nabla_{\theta} J = \mathbb{E}_{y \sim \pi_{\theta}} [r(x, y) \nabla_{\theta} \log \pi_{\theta}(y | x)]$$

Why is online needed?

- Why not just keep doing STaR?



	STaR (offline rounds)	Online RL
Samples from	An older model	The current model
Wrong answers	Discarded	Pushed down
Updates	A few large rounds	Every batch
Credit	Keep or drop	Relative to other samples

Online RL learns from failures and always trains on its own current distribution

o1 and R1

OpenAI o1 (Sep 2024)

- Long hidden CoT before answering
- Scaling with RL compute and test-time compute
- Closed: no recipe, reasoning not shown

DeepSeek R1 (Jan 2025)

- Open weights (MIT) and a paper with the recipe
- Matched o1 on AIME 2024 (79.8% vs. 79.2%)
- **R1-Zero**: RL on the base model alone, long CoT emerges
- Simple: GRPO + rule-based rewards, no reward model
- **What is GRPO?**

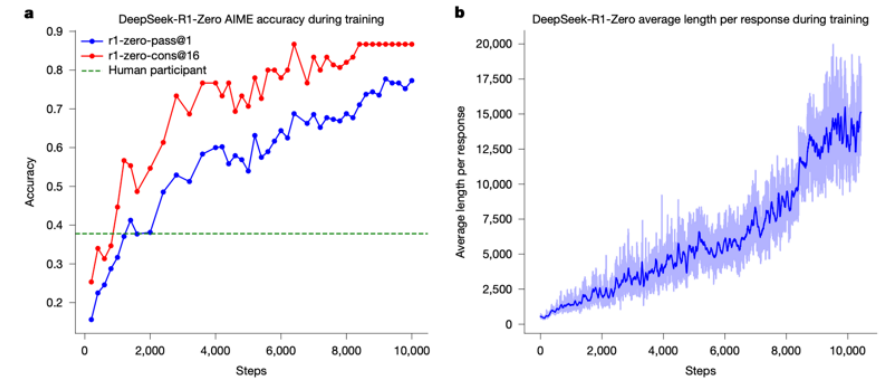


Fig. 1 | Accuracy and output length of DeepSeek-R1-Zero throughout the training process. **a.** AIME accuracy of DeepSeek-R1-Zero during training. AIME takes a mathematical problem as input and a number as output, illustrated in Extended Data Table 1. pass@1 and cons@16 are described in Supplementary Information, section 4.1. The baseline is the average score achieved by human

participants in the AIME competition. **b.** The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time. Note that a training step refers to a single policy update operation.

Recall: PPO

$$J_{\text{PPO}}(\theta) = \mathbb{E} \left[\frac{1}{|y|} \sum_t \min(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1 - \varepsilon, 1 + \varepsilon) \hat{A}_t) \right] - \beta D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})$$
$$\rho_t = \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}$$

- **Advantage** $\hat{A}_t$: was token t better than expected?
- **Ratio** ρ_t : how much the update changed that token's probability
- **Clipping**: no extra gain once ρ_t leaves $[1 - \varepsilon, 1 + \varepsilon]$ (usually 0.8-1.2)
- $\hat{A}_t$ comes from a learned **value model** V_ψ (GAE)
 - Value model learns: what is the value of this state?

The problem GRPO solves



- **Value model:** about the size of the policy, so roughly 2× memory and compute
- One reward at the end makes a per-token value hard to learn
- **GRPO** (Shao et al., 2024): no value model; baseline = mean reward of G samples per prompt

GRPO: Group-Relative Policy Optimization

1. For prompt x , sample G completions $y_1, \dots, y_G \sim \pi_{\theta_{\text{old}}}$
2. Score each: $r_i = r(x, y_i)$
3. **Group-relative advantage** (below), shared by every token of y_i
4. Update with PPO's clipped objective + KL to π_{ref}

$$\hat{A}_i = \frac{r_i - \text{mean}(r_1, \dots, r_G)}{\text{std}(r_1, \dots, r_G)}$$

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y_i \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_t \min(\rho_{i,t} \hat{A}_i, \text{clip}(\rho_{i,t}, 1 - \varepsilon, 1 + \varepsilon) \hat{A}_i) \right] - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}})$$

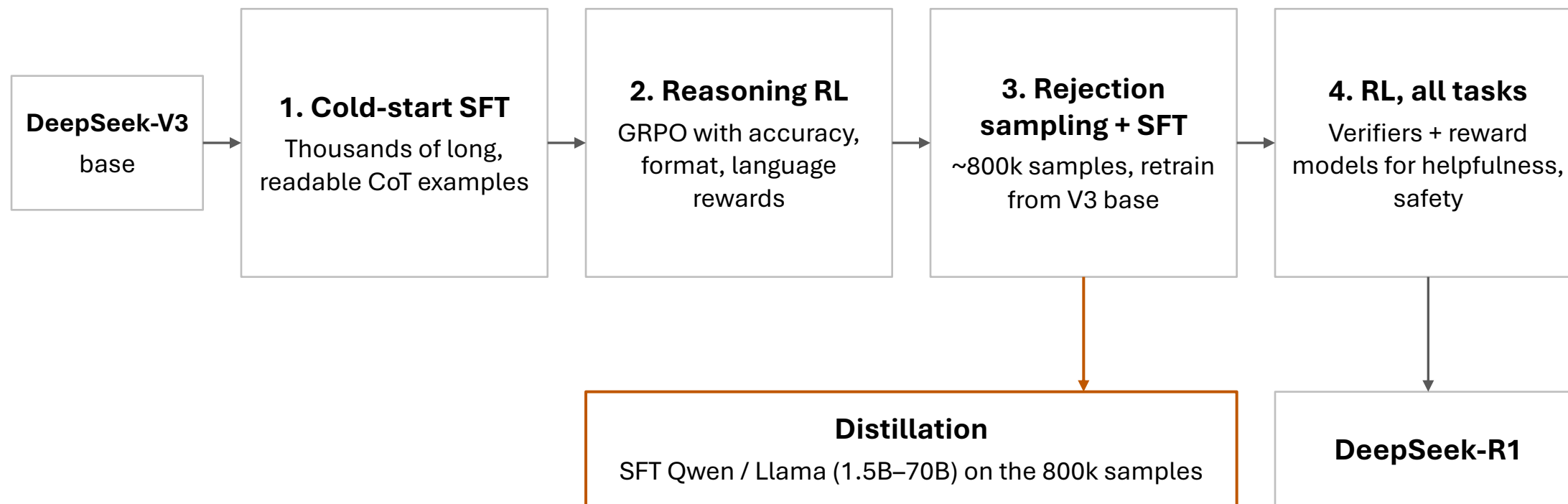
GRPO example

$G = 4$ samples per prompt, binary verifier reward

Prompt	Rewards	Mean	Std	Advantages
Half solved	1, 0, 0, 1	0.50	0.50	+1, -1, -1, +1
Rarely solved	1, 0, 0, 0	0.25	0.43	+1.73 , -0.58, -0.58, -0.58
Always solved	1, 1, 1, 1	1.00	0.00	0, 0, 0, 0
Never solved	0, 0, 0, 0	0.00	0.00	0, 0, 0, 0

- A rare success gets the **biggest advantage** (+1.73)
- All-correct or all-wrong groups: $\hat{A}_i = 0$, **no gradient**
 - “Dead groups”, usually filtered out to avoid wasted gradient steps

DeepSeek R1 recipe



Two RL stages, two SFT stages, and a retrain from the base model

More standard recipe

- Pre-train
- Mid-train
- Instruction tune
- RLHF
- RLVR

Reasoning models

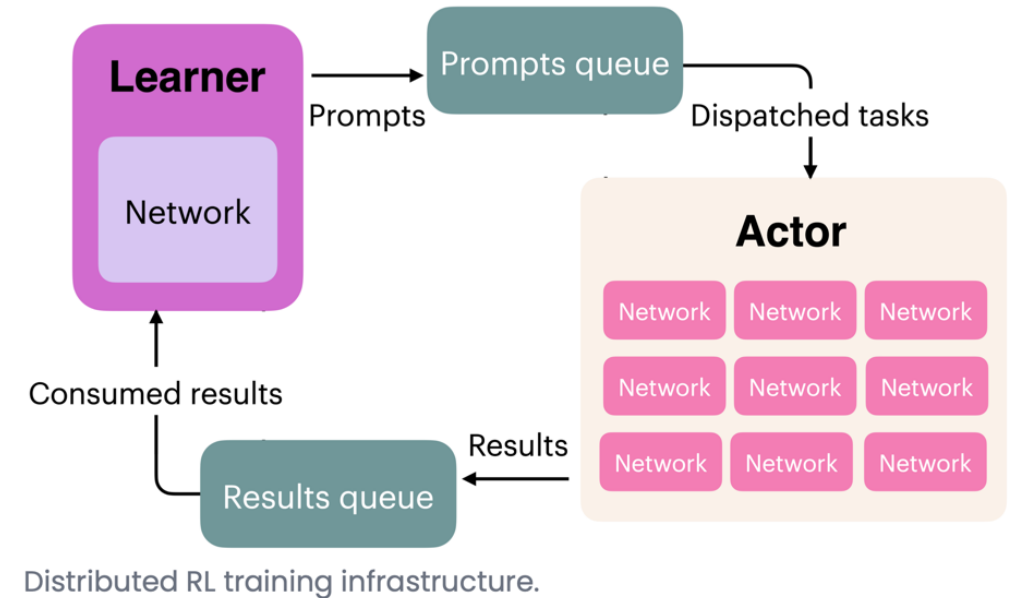
Model	Released	Notable
DeepSeek R1	Jan 2025	R1-Zero: RL alone; open recipe
Kimi 1.5	Jan 2025	Curriculum and length control; no open weights
Open-Reasoner-Zero	Feb 2025	Fully open replication with vanilla PPO
Qwen 3	Apr 2025	Hybrid: /think and /no_think modes
MiniMax-M1	Jun 2025	CISPO loss; FP32 LM head fix
GLM-4.5	Jul 2025	Reasoning plus agentic tool use
Olmo 3 Think	Nov 2025	Fully open: data, code, checkpoints
DeepSeek V3.2	Dec 2025	Thinking inside tool calls

Closed: o1/o3, Gemini 2.5, Claude. **2026:** Kimi K2.6, GLM-5.1, DeepSeek V4, Nemotron 3 Super

Source: Lambert, RLHF Book course, Lecture 5 (The rise of reasoning models)

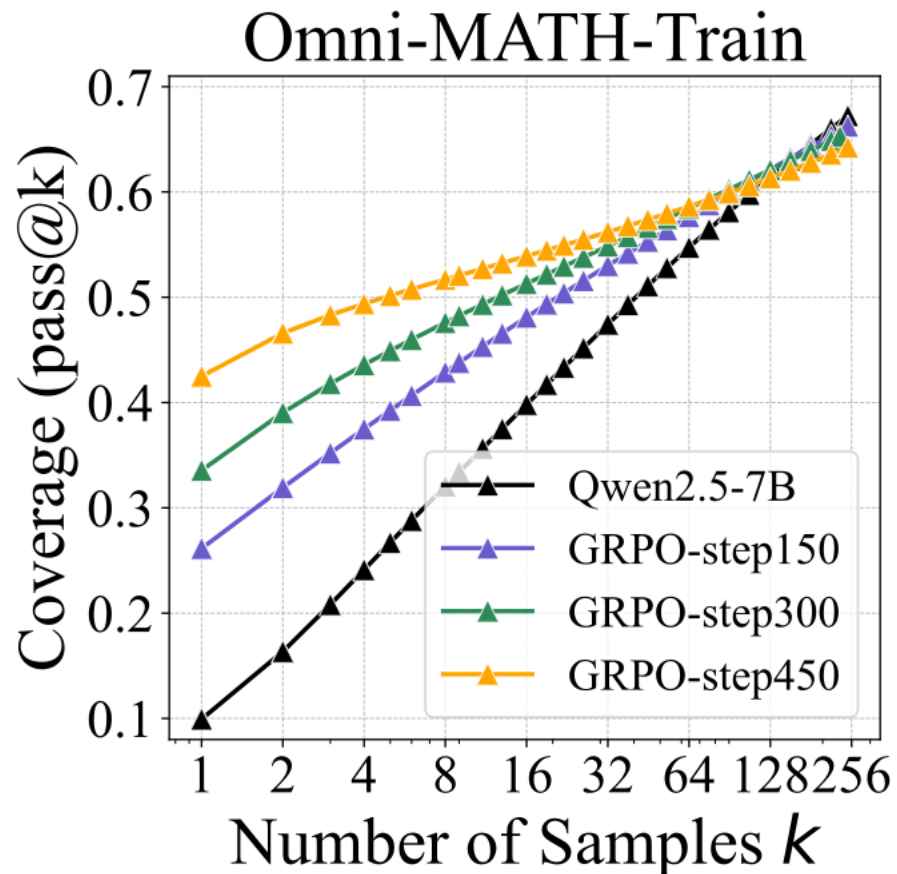
Challenge: inference is the bottleneck

- Rollouts are long and uneven; the learner has to wait for the slowest rollout
- Olmo 3: learner GPUs **idle ~75%**, 5–14× more inference than training compute
- **Async / off-policy**: actors generate nonstop, the learner trains on what arrives
- Cost: samples come from a slightly stale $\pi_{\theta_{\text{old}}}$; the ratio ρ_t and clipping correct for it



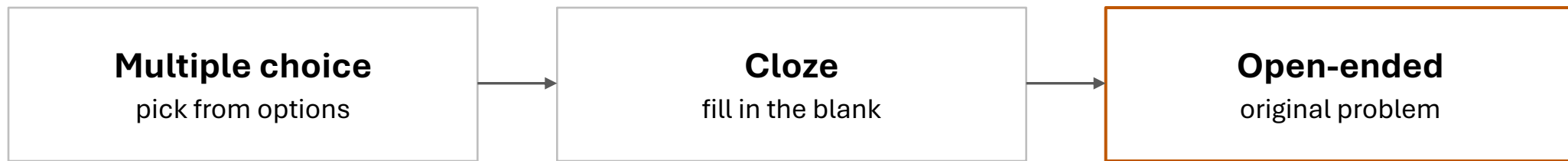
Can RL go past pass@k?

- Compare base vs. RLVR model on pass@ k for large k (Yue et al., 2025)
- Small k : **RL model wins**
- Large k : **base model wins**
- RLVR sharpens solutions the base can already sample
 - What about this the model cannot do?
- Open question: **does longer RL training push past it?**
 - Is it just sharpening the distribution to the best of k ?
 - Or is it improving iteratively?



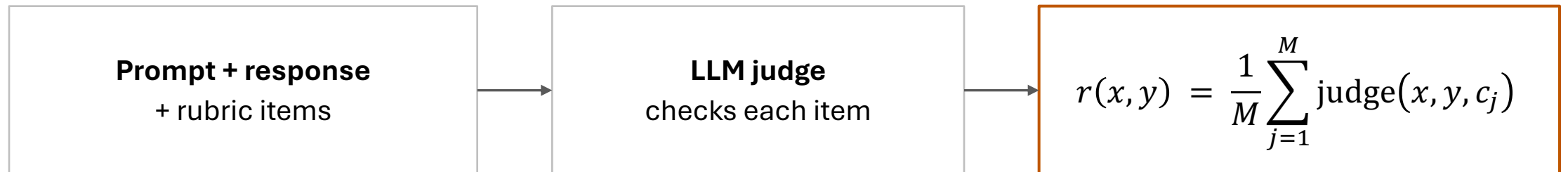
What if pass@k is zero?

- If no sample is correct: every $r_i = 0$, so $\hat{A}_i = 0 \rightarrow$ **no gradient**
- These are the hard problems we most want to learn from
- **Cog-DRIFT** (Chen et al., 2026): rewrite into easier formats, same answer
- Curriculum back to open-ended: **+10.1 (Qwen), +8.6 (Llama)** points on unsolved problems



What about non-verifiable tasks?

- No answer key: essays, advice, natural-language proofs
- **LLM-as-a-judge**: a strong LLM scores the response; the score becomes $r(x, y)$
- **Rubrics**: split quality into checkable items, judge each yes/no
- **Risk**: judge biases (length, position, self-preference) become targets. Reward hacking becomes easier



Reward hacking

The policy optimizes the reward you wrote, **including its flaws**

Setting	What the model learned
Boat racing game	Circle forever hitting the same targets instead of finishing (CoastRunners, 2016)
Coding with unit tests	Edit or bypass the tests instead of fixing the code (Baker et al., 2025)
LLM judge	Jailbreaks or exploiting of biases
Learned reward model	Longer answers score higher, so outputs get verbose (Singhal et al., 2023)

Defenses: KL to π_{ref} , stronger verifiers, held-out checks, monitoring the CoT

CoT monitoring

- What are the pros of CoT monitoring?
- What are the potential risks?

RLVR datasets and environments

Domain	Standard sources	Verifier
Math	GSM8K, MATH, NuminaMath, DeepScaleR, DAPO-Math	Answer match (Math-Verify)
Code	APPS, CodeContests, TACO; SWE-Gym for repo-level tasks	Unit tests in a sandbox
Other	Tülu 3 checkable instructions, logic puzzles	Constraint or answer check
Synthetic	Reasoning Gym, Enigmata: generated puzzles, any difficulty	Built-in checker

Scale: tens of thousands to ~1M prompts, reused over many epochs and filtered by difficulty

Takeaways

- Reasoning = more tokens spent on intermediate steps
- Sample + verify works, but is **costly at test time**
- RLVR moves that search into training: **pass@1** → **pass@k**
- **GRPO**: PPO without a value model, group mean as baseline
- Open: inference cost, $\text{pass}@k = 0$, going past $\text{pass}@k$, non-verifiable rewards