TEXAS
The University of Texas at Austin

# VERIFIABLE REWARDS AND GRPO

2026F INF385T: Post-Training of LLMs | Week 11

**CHAN YOUNG PARK**
Assistant Professor, iSchool, School of Computing, The University of Texas at Austin

# Announcements

- HW2 is due next Thursday (Nov 12)
- Next week's class (Nov 11)
  - Concept check quiz
  - Topics posted on canvas
  - Bring a pen!!

# Project logistics for the remaining classes

- **Nov 4: Project checkpoint presentation**
- Nov 18: Final project presentation
- Dec 2: Feedback-based revision
- Dec 7: Final report and code due

# Recap

**Week 8: Preferences→ Reward model, PPO**

**Week 9: Preference pairs → DPO**

**Week 11:Checkable answers → RLVR, GRPO**

# Week 11 overview

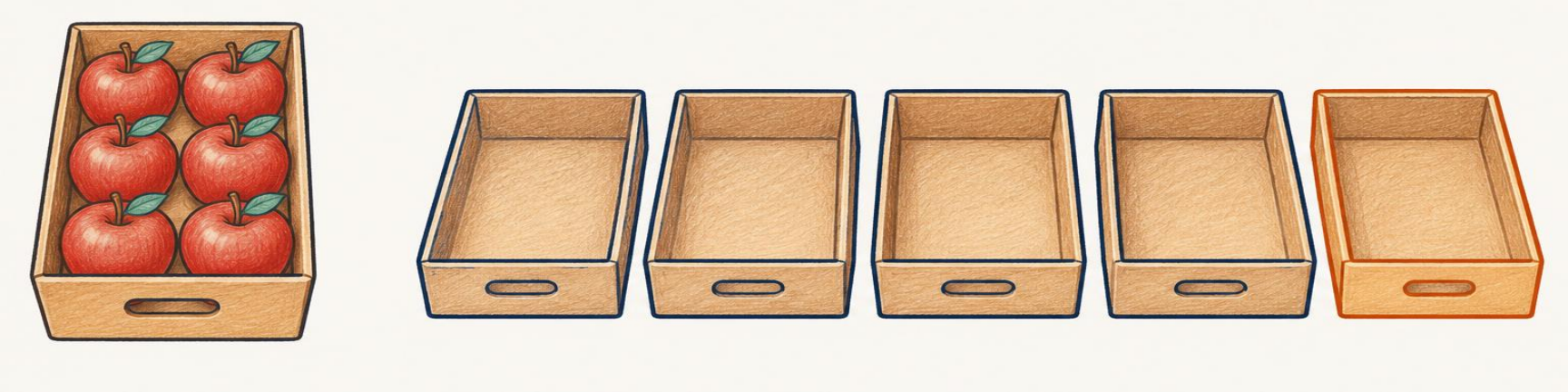
- **Sample and verify**
  Why move the search from test time into training? (Wang et al., 2023)
- **Verifiable rewards**
  When can a program replace the reward model? (Lambert et al., 2024; DeepSeek-AI, 2025)
- **GRPO**
  How do we drop the critic? (Shao et al., 2024)
- **In practice**
  What breaks and what does it teach? (Yu et al., 2025; Yue et al., 2025)

Why train with RL at all?

# SAMPLE AND VERIFY

# One question, many possible answers

You have 27 apples. Each box can hold up to 6 apples.
How many boxes do you need?



# Majority vote vs. verification

27 ÷ 6 = 4 r 3 → \answer{4}

27 ÷ 6 = 4.5 → \answer{4.5}

6 × 4 = 24 < 27, +1 box → \answer{5}

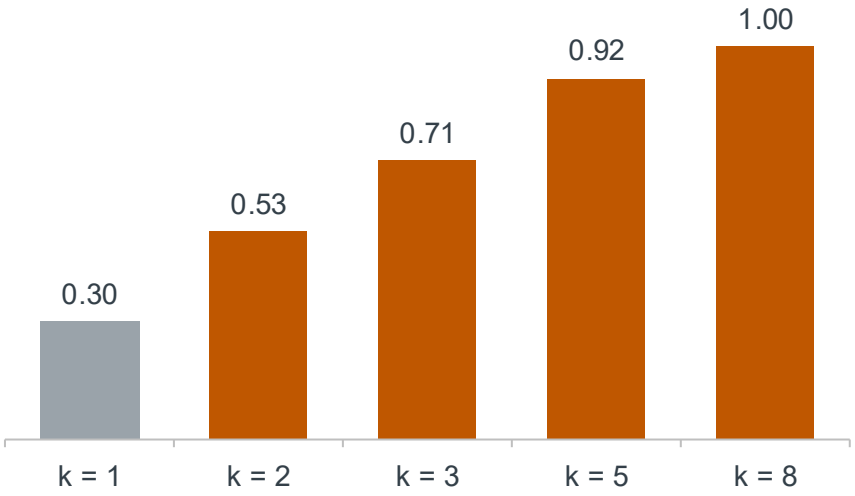
27 ÷ 6 = 4 r 3 → \answer{4}

**Majority vote: 4** **Self-consistency** (Wang et al., 2023)

**Verifier selects: 5** **Best-of-N with a checker**

A repeated answer can still be wrong

# pass@1 vs. pass@k




- pass@1: a single sample is correct
- pass@k: at least one of k is correct

pass@k measures coverage: what the model can solve at all

The gap between pass@1 and pass@k is what RLVR tries to close

# Why RL here?

- Test-time is costly: N samples = N× tokens per query
- No verifier (answer key) at test time

RLVR moves pass@1 toward pass@k

Run the search during training where answers are known, train on verified successes, and only generate one sample at test time

When can a program be the judge?

# VERIFIABLE REWARDS

# Tülu 3: Pushing Frontiers in Open Language Model Post-Training

Nathan Lambert[♥1,*] Jacob Morrison[♥1] Valentina Pyatkin[♥1,2] Shengyi Huang[♥1] Hamish Ivison[♥1,2]
Faeze Brahman[♥1] Lester James V. Miranda[♥1]

Alisa Liu[2] Nouha Dziri[1] Xinxi Lyu[1] Yuling Gu[1] Saumya Malik[1] Victoria Graf[2] Jena D. Hwang[1]
Jiangjiang Yang[1] Ronan Le Bras[1] Oyvind Tafjord[1] Chris Wilhelm[1]

Luca Soldaini[1] Noah A. Smith[1,2] Yizhong Wang[1,2] Pradeep Dasigi[1] Hannaneh Hajishirzi[1,2]
[1]Allen Institute for AI, [2]University of Washington

*Tülu 3 was a team effort. ♥ marks core contributors. See full author contributions here.

Lambert et al. (2024), arXiv:2411.15124

# Reward model vs verifier

| | Reward model | Verifier |
|---|---|---|
| **What it is** | a network trained on preferences | a program (parse + compare), run tests |
| **Output** | any real number | 0 or α (Tülu 3: α = 10) |
| **Weak spots** | length, style, flattery | format mismatches, narrow checks |
| **Coverage** | any prompt | only checkable tasks |

Same objective as reward model from week 8 (max E[reward] − β·KL)
Only the source of reward changes

# What counts as verifiable?

| Domain | Verifier | A pass tells you |
|---|---|---|
| **Math** | extract the final answer, compare to y* | the final answer matches (not the reasoning) |
| **Code** | run unit tests in a sandbox | the code passes those tests |
| **Precise IF** | constraint functions (IFEval-style) | listed constraints met (“exactly 3 bullets”) |
| **Format** | regex: <think> … </think> then answer | structure only |

Verifiable = cheap, deterministic, but narrow

# The verifier can be wrong too

**False negative: right, reward 0**

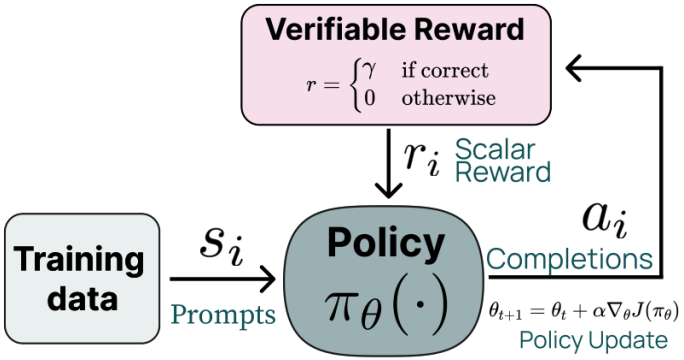
- “The answer is five boxes.” (no \answer{})
- “5.0” vs. “5”

**False positive: wrong, reward 1**

- Right answer from wrong reasoning or a guess
- Multiple choice: 25% by chance

Verifier defines the task (at least for RLVR)
A verifier is essentially a rubric written in code
audit it like a judge

# Tülu 3: RLVR as the final stage

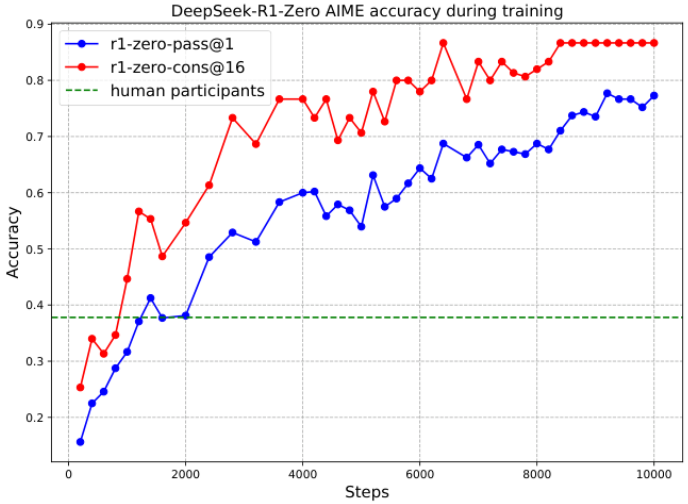



**SFT → DPO → RLVR**

- ~30K prompts from GSM8K, MATH, IFEval
- PPO, α = 10
- Gains on the targeted skills, not on everything
- Over-optimization even with a ground-truth checker
- Checker-only reward beat adding RM scores

Lambert et al. (2024). Tülu 3: Pushing Frontiers in Open Language Model Post-Training

# DeepSeek-R1-Zero: RLVR from a base model




- No SFT
- GRPO, accuracy + format rewards only
- **AIME pass@1 15.6% → 77.9%**

  A verifier plus a strong base model was enough to elicit long reasoning

RLVR = where r comes from, not how we update
(Tülu 3: PPO; R1: GRPO)

DeepSeek-AI (2025). DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning (Nature)

How does a group replace the critic?

# GRPO

deepseek

# DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

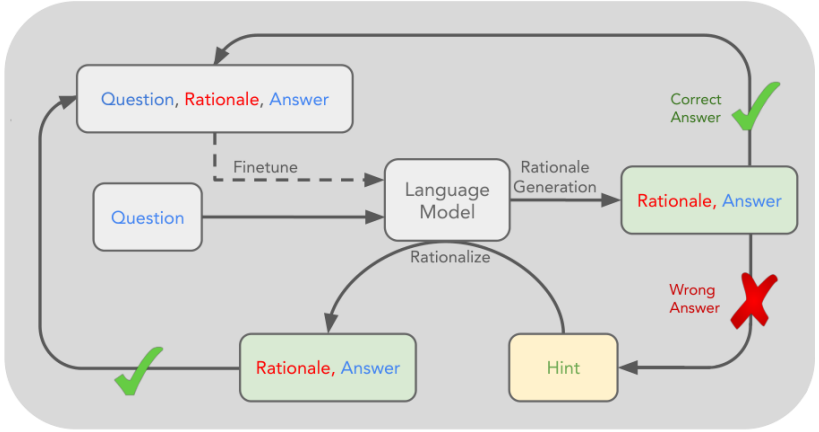
Zhihong Shao[1,2*†], Peiyi Wang[1,3*†], Qihao Zhu[1,3*†], Runxin Xu[1], Junxiao Song[1]
Xiao Bi[1], Haowei Zhang[1], Mingchuan Zhang[1], Y.K. Li[1], Y. Wu[1], Daya Guo[1*]

[1]DeepSeek-AI, [2]Tsinghua University, [3]Peking University

{zhihongshao,wangpeiyi,zhuqh,guoday}@deepseek.com
https://github.com/deepseek-ai/DeepSeek-Math

Shao et al. (2024), arXiv:2402.03300

# STaR: keep what verifies, fine-tune, repeat




```
Q: What can be used
to carry a small dog?
Answer Choices:
(a) swimming pool
(b) basket
(c) dog show
(d) backyard
(e) own home
A: The answer must be
something that can be
used to carry a small
dog. Baskets are
designed to hold things.
Therefore, the answer
is basket (b).
```

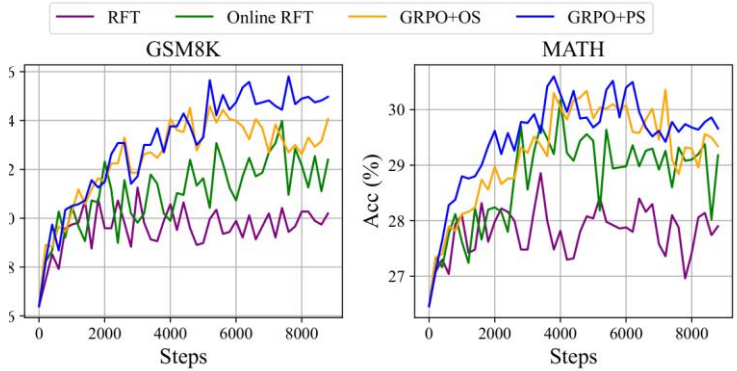
generate → check → keep correct rationales → fine-tune

Binary-reward filtering (=rejection sampling fine-tuning aka RFT) is already a policy gradient

Zelikman et al. (2022). STaR: Bootstrapping Reasoning With Reasoning (NeurIPS)

# Why go online?

| | STaR (offline) | Online RL |
|---|---|---|
| **Samples from** | an older model | the current model |
| **Wrong answers** | discarded | pushed down |
| **Updates** | a few big rounds | every batch |
| **Credit** | keep or drop | vs. other samples |

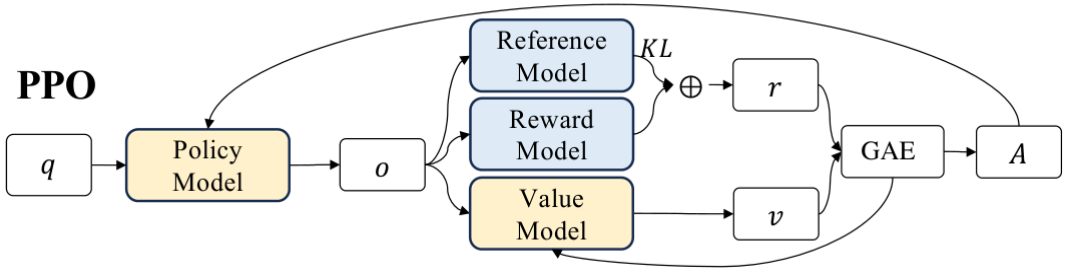



DeepSeekMath shows GRPO > Online RFT > RFT

"Relative to other samples" needs a baseline
PPO learns one (critic) vs GRPO computes one

Shao et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

# Recall: PPO

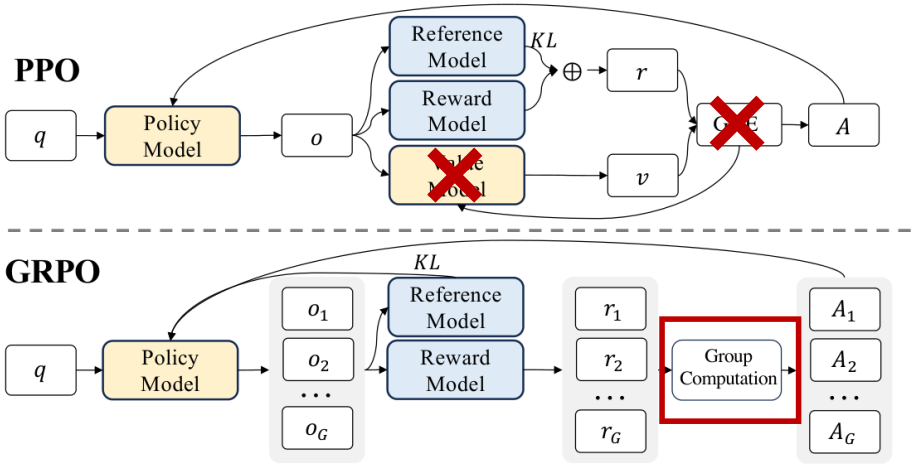



$$J_{\text{PPO}}(\theta) = \mathbb{E}\left[\frac{1}{|y|}\sum_t \min\left(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right] - \beta D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})$$

$$\rho_t = \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}$$

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

# GRPO: Group Relative Policy Optimization

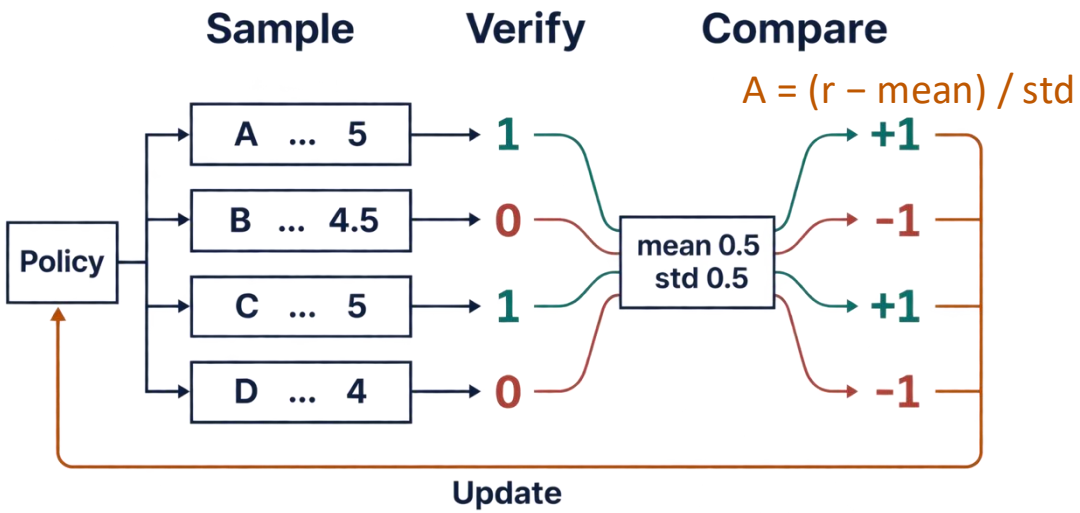



$$\hat{A}_i = (r_i - \text{mean}) / \text{std}$$

- mean and std over G samples of the same prompt
- the group mean plays the critic's role
- one advantage per response

Shao et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

# GRPO loop




(1) sample a group for the same prompt
(2) verify each response
(3) compare each reward with its group

- One advantage per response
  - Push up every token of A and C
  - push down every token of B and D

# Same reward can carry different weight

**One success in the group**

1 0 0 0

Successful response

**A ≈ +1.73**

**Three successes in the group**

1 1 1 0

Successful response

**A ≈ +0.58**

**“Relative” depends on the other samples for that prompt**

# Which prompts teach anything?

| Group | Rewards | Advantages | Signal |
|---|---|---|---|
| **Too easy** | 1 1 1 1 | 0 0 0 0 | none |
| **Too hard** | 0 0 0 0 | 0 0 0 0 | none |
| **Mixed** | 1 0 1 0 | +1 −1 +1 −1 | yes |
| **Rare success** | 1 0 0 0 | +1.73 −0.58 −0.58 −0.58 | big push on the success |

Gradient only exists where the model sometimes succeeds
= GRPO learns at the edge of what the model can already do

In practice, “Dead groups” are usually filtered out to avoid wasted gradient steps

# The GRPO objective

$$\rho_t = \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}$$

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

| | Advantage | KL to π_ref |
|---|---|---|
| **PPO** | critic V + GAE (per token) | per-token fee inside the reward |
| **GRPO** | group-normalized (per response) | loss term (often β = 0 now) |

Same clipped update as PPO
only the advantage changes

# GRPO in code

```
# 1. Rollout: G samples per prompt from the current policy
out = policy.generate(prompts, do_sample=True, num_return_sequences=G)
r = verify(decode(out), answers).view(B, G)      # 0/1 from a checker

# 2. Group-relative advantage (no critic)
adv = (r - r.mean(1, keepdim=True)) / (r.std(1, keepdim=True) + 1e-4)
adv = adv.view(B * G, 1)                          # same for every token

# 3. Clipped policy-gradient update (PPO-style)
ratio = torch.exp(logp_new - logp_old)            # [B*G, T]
loss = -torch.min(ratio * adv, ratio.clamp(1 - eps, 1 + eps) * adv)
loss = ((loss * mask).sum(1) / mask.sum(1)).mean()  # per-response mean
```

TRL's GRPOTrainer wraps this loop

# What GRPO + RLVR remove

**Policy $\pi_\theta$**
**player**
**trained**

**Value $V_\psi$**
**commentator**
**→ group mean**

**Reference $\pi_{ref}$**
**anchor**
**frozen (often dropped)**

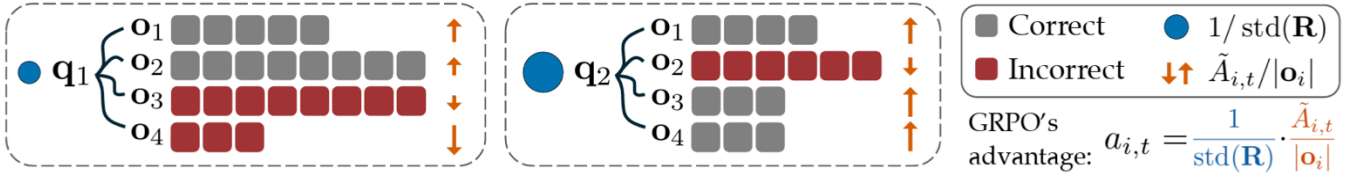
**Reward $r_\varphi$**
**judge**
**→ a program**

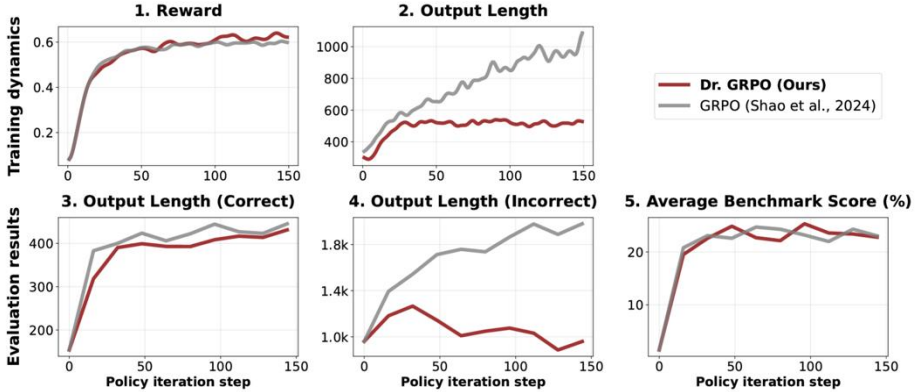
- Back to sampling at every step (unlike DPO): G rollouts per prompt
- One trained model (+ an optional frozen reference) in memory
- Cost moves to generation: long traces × G samples

What breaks, and what does it teach?

# GRPO IN PRACTICE

# Bias: long wrong answers are punished less




**Dr. GRPO fix**

$\hat{A}_i = r_i - \text{mean}$

normalize by a constant, not $|o_i|$ or std

- Gray (GRPO): wrong answers grow to ~2k tokens
- Red (Dr. GRPO): flat, same accuracy

Before calling longer traces "reasoning", check length on wrong answers

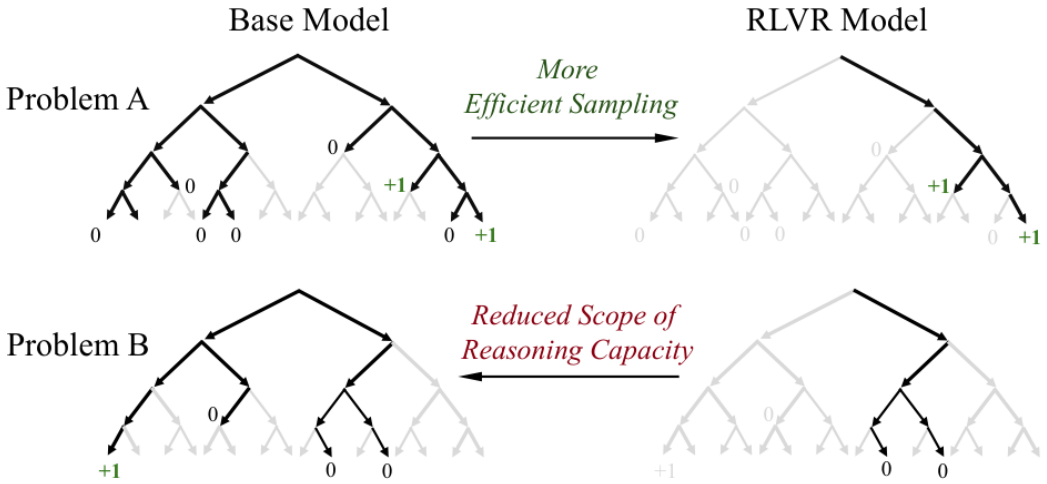
Liu et al. (2025). Understanding R1-Zero-Like Training: A Critical Perspective (COLM)

# What to monitor during GRPO

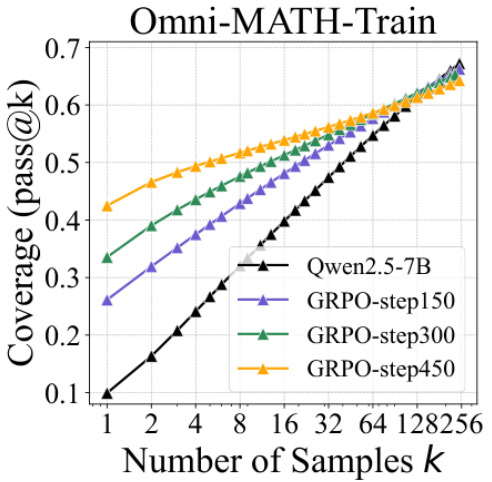
| Metric | Healthy | Warning sign |
|---|---|---|
| **Train reward** | Rises then plateaus | jumps to ~1 fast (too easy or hacked) |
| **Zero-variance groups** | a modest fraction | most groups all-0 or all-1 |
| **Response length** | grows with difficulty | grows on wrong or truncated answers |
| **Entropy** | slow decline | collapse → all G samples identical |
| **Held-out pass@1 / pass@k** | pass@1 up, pass@k flat or up | pass@k drops (narrowing) |
| **Parse / format rate** | near 100% early | gains come only from format |

Read samples with reward 1 (not just the curves)

# Can RL go past pass@k?



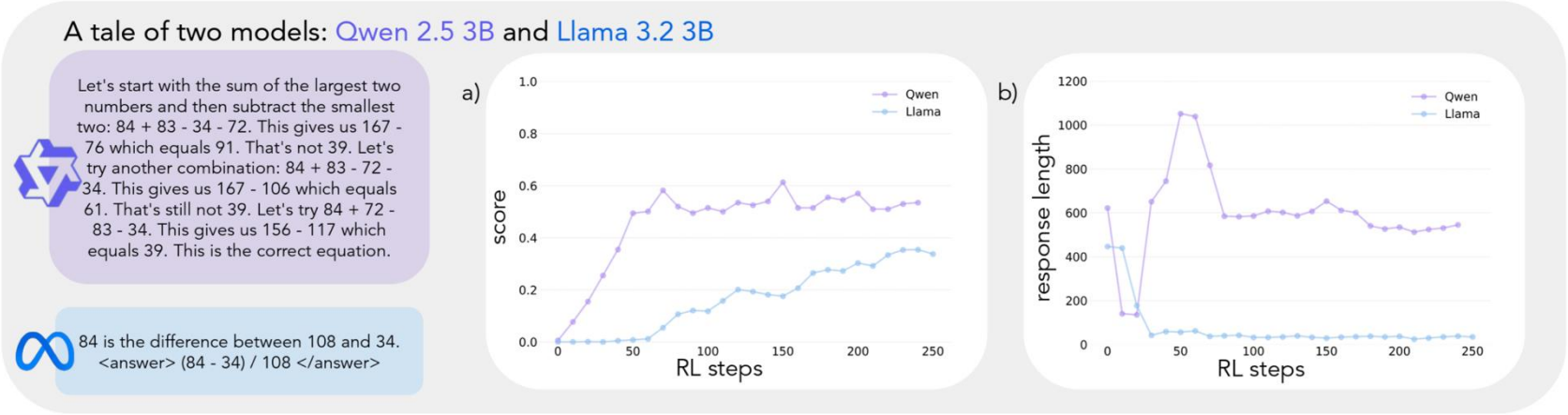

RLVR wins at k = 1
The base model wins at large k




RLVR mostly sharpens
Report pass@1 and pass@k

Yue et al. (2025). Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? (NeurIPS)

# RL amplifies existing behaviors




- Qwen 2.5 3B already verifies and backtracks → RL climbs fast
- Llama 3.2 3B does not → RL stalls
- Priming Llama with these behaviors (short SFT) closes most of the gap

RL selects among behaviors the model can already produce

Gandhi et al. (2025). Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs

# Should your project use RLVR?

1 **A useful verifier**

Audit it on correct and wrong answers

2 **Occasional success**

Sample from the starting model before training

3 **A rollout budget**

Estimate tokens and fix a held-out evaluation

RLVR sharpens what the model can already sometimes do

# Conclusion

- RLVR moves sample-and-verify from inference into training (pass@1 → pass@k)
- A verifier (program) replaces the learned RM
- GRPO: using a group baseline $\hat{A} = (r - \text{mean}) / \text{std}$ to compute advantages
    - Only prompts the model sometimes solves give a gradient
    - Rollouts dominate cost
- Before claiming “RL taught reasoning” monitor pass@k, length, another model family