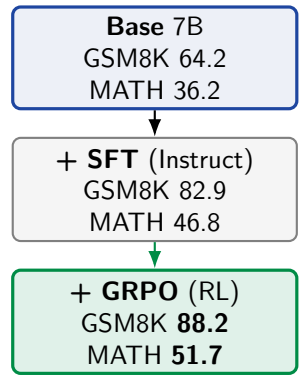
A 7B model that does competition math

DeepSeekMath-RL 7B scores **51.7%** on the MATH benchmark and **88.2%** on GSM8K – with chain-of-thought only, no external tools.

That beats every open 7B–70B model of its time and approaches GPT-4 (52.9%) and Gemini-Ultra (53.2%).

The last training stage that got it there was **reinforcement learning**. The RL algorithm they introduced – **GRPO** – is now the workhorse behind DeepSeek-R1 and much of the "reasoning model" wave.

Today: *what GRPO is, why it drops PPO's most expensive piece, and why it works.*



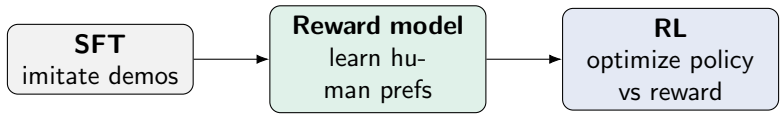
Outline

Background: how do we RL-tune an LLM?

Before GRPO can look clever, we need PPO and its price tag.

Where RL sits in the pipeline

The alignment recipe (Ouyang et al., 2022) has three stages after pretraining:



DeepSeekMath plugs in at the **last** stage: start from the SFT model (DeepSeekMath-Instruct), then RL against a reward model.

The policy is a language model. An "action" is emitting the next token; a "trajectory" is a full generated answer o to a question q.

PPO: the standard RL objective

PPO maximizes reward while staying close to the sampling policy via a **clipped** importance ratio:

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{q,\, o\sim\pi_{\theta_{\text{old}}}} \frac{1}{|o|} \sum_{t=1}^{|o|} \min[\rho_t\, A_t,\ \text{clip}(\rho_t,\, 1-\varepsilon,\, 1+\varepsilon)\, A_t]\,, \qquad \rho_t = \frac{\pi_\theta(o_t \mid q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q, o_{<t})}$$

- ρ_t – how much more likely token o_t is under the new vs old policy.
- A_t – the **advantage**: was this token better or worse than expected?
- clip – forbid steps that move the ratio too far, for stability.

Everything hinges on A_t. Where does the "expected" baseline come from? That is the job of a second network – the **critic**.

The critic (value network) and its cost

To estimate A_t, PPO trains a **value model** V_ψ that predicts expected return, and computes A_t by **GAE** from per-token rewards and V_ψ.

Subtracting a baseline reduces gradient *variance* without adding bias – that is why we want V_ψ.

But the price is steep:

- V_ψ is "another model of comparable size to the policy" – a second 7B network, trained *and* stored.
- The reward usually lands **only on the last token**, yet V_ψ must be accurate at *every* token.

PPO keeps **four** models live:

1. Policy π_θ (trained)
2. Reference π_{ref} (frozen)
3. Reward r_φ
4. **Value V_ψ (trained, big)**

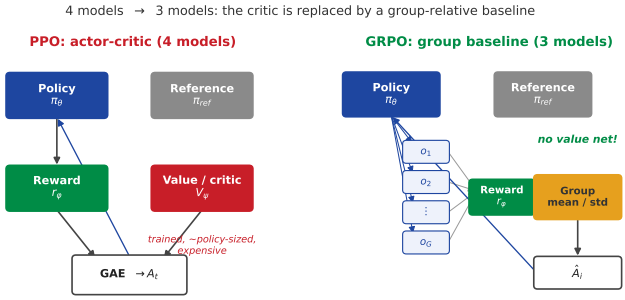
The value net is the expensive one we would love to delete.

What if the baseline came from a group, not a critic?

Sample several answers to the same question; judge each against their average.

Group sampling replaces the value net

For each question q, sample a **group** of G outputs $\{o_1, \ldots, o_G\} \sim \pi_{\theta_{\text{old}}}$, score them with the reward model, and use the group's own statistics as the baseline.

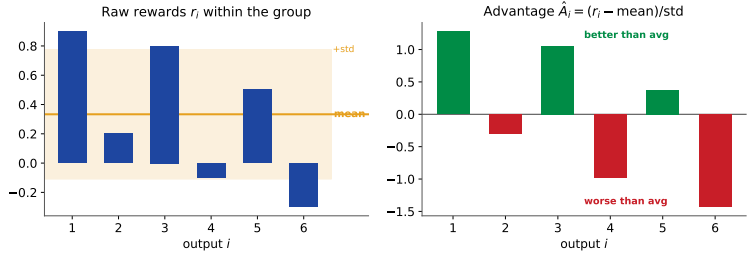


No value network. The baseline is the group mean – and this matches how reward models are trained: on **comparisons** between answers to the same prompt.

Group-relative advantage (outcome supervision)

Reward model returns one score per output, $\mathbf{r} = \{r_1, \ldots, r_G\}$. Normalize within the group, then hand that scalar to *every* token of the output:

$$\tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}, \qquad \hat{A}_{i,t} = \tilde{r}_i \quad \text{for all } t$$



Positive advantage = "better than the group average, push these tokens up." Negative = "worse, push down." *(rewards shown are illustrative.)*

The GRPO objective

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\,\{o_i\}\sim\pi_{\theta_{\text{old}}}} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big\{\min\Big[\rho_{i,t}\hat{A}_{i,t},\ \text{clip}(\rho_{i,t},1-\varepsilon,1+\varepsilon)\hat{A}_{i,t}\Big] - \beta\,\mathbb{D}_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}]\Big\}$$

Two changes vs PPO – both worth saying out loud:

- $\hat{A}_{i,t}$ comes from the **group**, not from V_ψ. The critic is gone.
- The KL term is added **directly to the loss** (coefficient β), not folded into the per-token reward as PPO does. This keeps the advantage clean.

$\pi_{\theta_{\text{old}}}$ (behavior policy, generated the samples) and π_{ref} (frozen KL anchor) are *different* models. Don't merge them.

Two details that matter

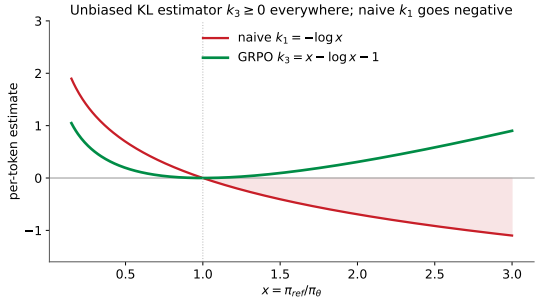
The KL estimator, and per-step credit assignment.

The unbiased KL estimator

GRPO does *not* use the naive $\log(\pi_\theta/\pi_{\text{ref}})$. It uses Schulman's estimator (with $x = \pi_{\text{ref}}/\pi_\theta$):

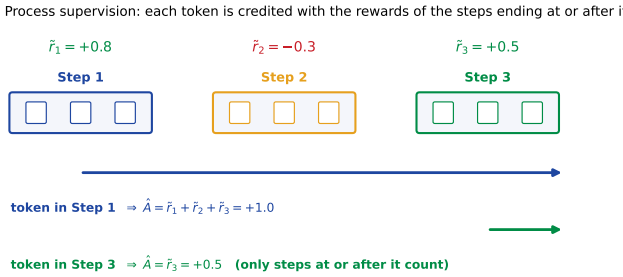
$$\mathbb{D}_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}] \approx \frac{\pi_{\text{ref}}}{\pi_\theta} - \log \frac{\pi_{\text{ref}}}{\pi_\theta} - 1 \;=\; x - \log x - 1$$

- **Unbiased**: its expectation is the true KL.
- **Always** ≥ 0: since $x - \log x - 1 \geq 0$.
- Lower variance than the naive log-ratio, which can go *negative* on a single sample.



Process supervision: per-step credit

Process supervision scores each reasoning *step*; a token's advantage sums the normalized rewards of steps ending at or after it – $\hat{A}_{i,t} = \sum_{\mathrm{idx}(j) \geq t} \tilde{r}_i^{\mathrm{idx}(j)}$, with $\tilde{r} = \big(r - \mathrm{mean}(R)\big) / \mathrm{std}(R)$:



Outcome vs process is just *how finely* the group-normalized reward is spread over the tokens – same idea, finer credit.

Putting it together

One iteration of GRPO, end to end.

Algorithm: iterative GRPO

For each step:

1. Set the behavior policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$.
2. For each question q in the batch, **sample** G **outputs** $\{o_i\} \sim \pi_{\theta_{\text{old}}}$.
3. **Score** each output with the reward model: $r_i = r_\varphi(q, o_i)$.
4. **Normalize** within the group $\rightarrow \hat{A}_{i,t}$ (group mean/std).
5. **Update** π_θ by ascending the GRPO objective (clipped ratio + direct KL).

Outer loop (what makes it iterative): periodically reset the reference model to the current policy, and **continually retrain the reward model** on fresh policy samples using a replay buffer (10% history).

Their production run does a *single* policy update per exploration step ($\mu = 1$), so $\pi_\theta \approx \pi_{\theta_{\text{old}}}$.
Hyperparameters: $G = 64$, $\beta = 0.04$, policy lr 10^{-6}.

GRPO vs PPO, side by side

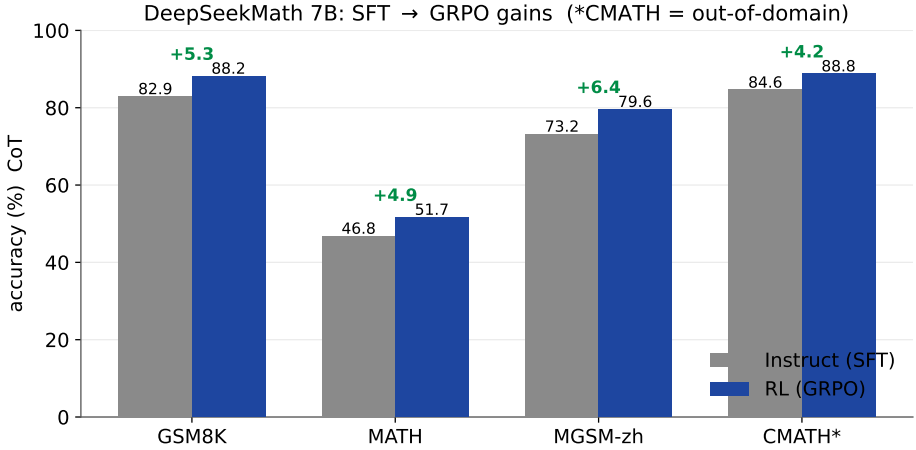
	PPO	**GRPO**
Baseline for A	learned value net V_ψ (big)	group mean / std
Models in memory	4 (incl. value)	3 (no value)
KL regularization	folded into the reward	direct in loss, unbiased estimator
Sampling	one output / prompt	**group of** G / prompt
Fit to reward model	–	matches its *comparative* training

Dropped: the value network + GAE. **Gained:** lower memory, a baseline matched to the reward model. **Cost:** you now generate G samples per prompt – critic-free is not compute-free.

Does it work?

The numbers, and a surprising diagnosis of *why*.

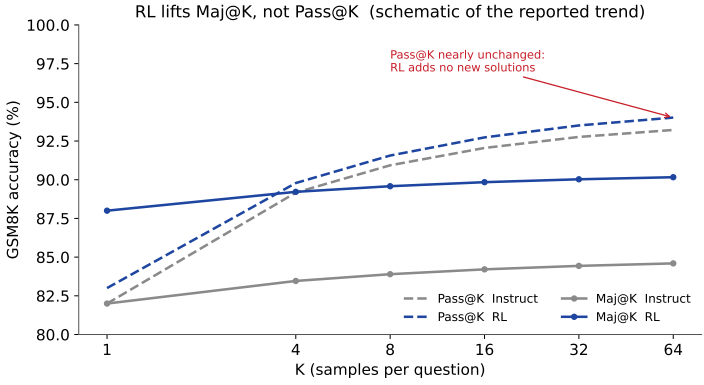
RL gains across benchmarks



RL used **only** GSM8K + MATH chain-of-thought questions – yet *every* benchmark improved, including out-of-domain CMATH (+4.2). Self-consistency over 64 samples pushes MATH to **60.9%**.

Predict-first: did RL make the model smarter?

Natural guess: RL taught the model *new* math it could not do before.



The paper's own analysis says **no**. RL lifts **Maj@K** but barely moves **Pass@K** (can *any* of K samples solve it). RL **sharpens** the distribution to surface answers the base model already had – it adds no new capability.

SFT, RFT, DPO, PPO, GRPO: one gradient

They differ only in *where data comes from* and *how reward becomes a coefficient*.

A unified view of LLM training

Every method the paper considers has the *same* gradient shape:

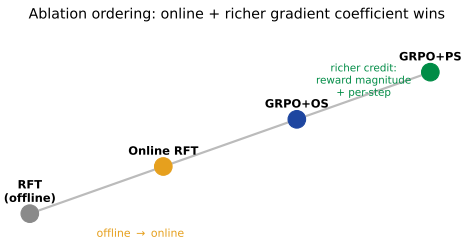
$$\nabla_\theta \mathcal{J}_{\mathcal{A}}(\theta) = \mathbb{E}_{(q,o)\sim\mathcal{D}} \frac{1}{|o|} \sum_{t=1}^{|o|} \underbrace{GC_{\mathcal{A}}(q,o,t)}_{\text{gradient coefficient}} \nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})$$

Three knobs fully characterize a method:

Method	**Data source**	**Reward**	**Grad. coefficient**
SFT	demos	–	1 (push every token up)
RFT	offline, from π_{sft}	rule	$\mathbb{I}$(correct)
Online RFT	online, from π_θ	rule	$\mathbb{I}$(correct)
PPO	online	model	A_t (GAE)
GRPO	online, *group*	model	$\hat{A}_{i,t}$ (group-relative)

Two axes explain the ranking: **online** > **offline** data, and **richer gradient coefficients** (real-valued reward, penalize wrong) beat binary correct/incorrect.

Why GRPO beats plain online RFT



- **RFT**: reinforce correct answers, offline. Never penalizes wrong ones.
- **Online RFT**: same, but sample from the live policy. Online helps.
- **GRPO**: scale each gradient by *how much better/worse than the group* – differentially reward *and* penalize.
- **+Process**: localize the credit to the good steps.

Ordering reproduces the paper's ablation; exact values are schematic.

Recap

- **GRPO = PPO without the critic.** Sample G answers per question; the group mean/std is the baseline.
- Advantage $\hat{A}_{i,t} = (r_i - \text{mean})/\text{std}$ – outcome or per-step (process) supervision.
- KL is added *directly to the loss* with an unbiased, non-negative estimator.
- Result: **4 models** $\rightarrow$ **3**, a baseline matched to the reward model, and state-of-the-art open math reasoning.
- RL here **sharpens** an existing distribution (Maj@K up, Pass@K flat) rather than adding new skills.
- All of SFT/RFT/DPO/PPO/GRPO are one gradient – differing in data source and gradient coefficient.

Next: *DPO* – skip the RL loop and the reward model entirely, and optimize preferences with a single classification-style loss.