GRPO: Learning from a Group of Answers

Why a relative reward can replace a learned critic

Fengnan Deng · September 2026

The idea in one sentence. Sample several answers to the same question, compare their rewards, and use that comparison to encourage better answers without training a separate value model.

1 From a learned critic to a group comparison

Supervised fine-tuning (SFT) teaches a language model to imitate demonstrations. Reinforcement learning (RL) lets it generate answers, receive rewards, and learn which behavior to repeat. This is attractive when checking an answer is easier than writing a good demonstration.

In an actor–critic method, the reward function evaluates an answer, while the critic predicts expected future return from a state, such as a partially written response. Its prediction helps determine whether a sampled action was better than expected. Training a large critic alongside the policy can be expensive.

PPO (2017) introduced a practical clipped objective [1]; InstructGPT (2022) combined SFT, a human-feedback reward model, and PPO [2]. DeepSeekMath (2024) introduced GRPO, replacing the learned value baseline with a comparison among answers to the same prompt [3]. DeepSeek-R1 later used GRPO in reasoning training; R1-Zero also showed that preliminary SFT is not required [4].

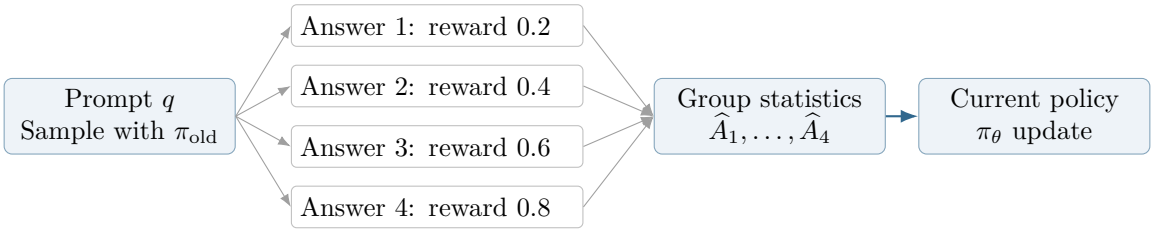

Figure 1: One prompt, four sampled answers, and a relative comparison that drives the update. Rewards are illustrative; the KL reference policy is omitted here.

How the comparison becomes an advantage

For a prompt q, sample $G \geq 2$ responses $o_i \sim \pi_{\text{old}}(\cdot \mid q)$. Let $R_i = R(q, o_i)$ be the reward and T_i the number of generated tokens. Using the population standard deviation, define

$$\bar{R} = \frac{1}{G}\sum_{j=1}^{G} R_j, \qquad \sigma_R = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(R_j - \bar{R})^2}, \tag{1}$$

$$\widehat{A}_i = \frac{R_i - \bar{R}}{\sigma_R + \delta}, \qquad \widehat{A}_{i,t} = \widehat{A}_i, \tag{2}$$

where $\delta > 0$ prevents division by zero. Under outcome supervision, all generated tokens in an answer share its advantage.

For Figure 1, $\bar{R} = 0.5$ and $\sigma_R \approx 0.224$. Neglecting δ, the advantages are $(-1.342, -0.447, +0.447, +1.342)$. Positive advantages encourage the sampled tokens; negative ones reverse that training contribution. This gives a relative learning signal without identifying which individual reasoning step was responsible for the reward.

One edge case matters: if all rewards are equal, every advantage is zero and the group supplies no reward-driven gradient. GRPO saves critic training, but still pays for multiple rollouts and their evaluation.

2 The objective: clipping and a reference policy

Three policy roles must remain distinct. The **current policy** π_θ is optimized. The **old policy** π_{old} generated the rollouts and stays fixed during their reuse. The **reference policy** π_{ref} is a frozen anchor for regularization, often the starting checkpoint for an RL stage.

Define the token probability ratio and clipped contribution by

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\text{old}}(o_{i,t} \mid q, o_{i,<t})}, \tag{3}$$

$$C_\epsilon(\rho, A) = \min\{\rho A, \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A\}\,. \tag{4}$$

An outcome-supervised GRPO objective is [3]

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\{o_i\}\sim\pi_{\text{old}}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\left(C_\epsilon(\rho_{i,t}, \widehat{A}_i) - \beta\widehat{d}^{\text{KL}}_{i,t}\right)\right]. \tag{5}$$

Prompts come from the training distribution. Training *maximizes* J; a loss-minimizing implementation uses $\mathcal{L} = -J$. The old log probabilities, rewards, and advantages are treated as constants. The averages give each response equal total normalization weight, regardless of its length.

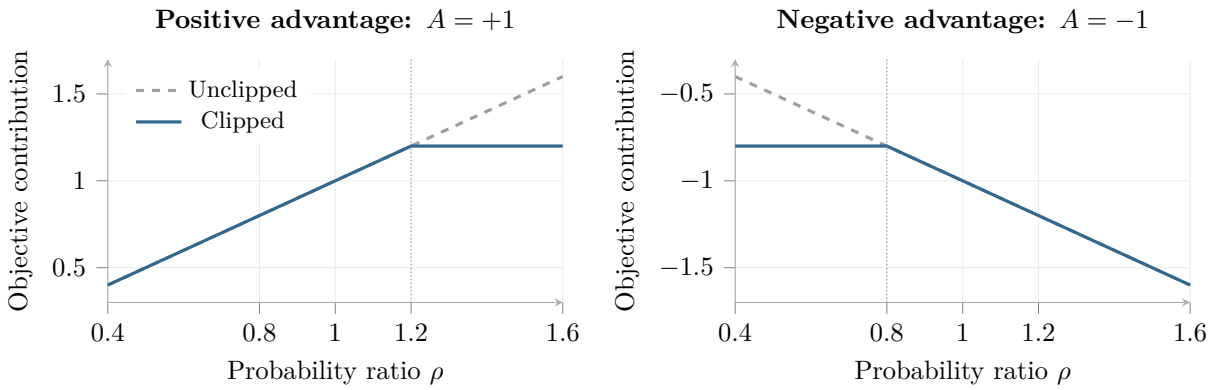

Figure 2: The PPO-style surrogate in Equation (4), with illustrative $\epsilon = 0.2$. Higher is better in both panels. Clipping flattens only the side that would otherwise reward a larger change in the favored direction.

For $A > 0$, increasing ρ beyond $1+\epsilon$ brings no further gain from this term. For $A < 0$, decreasing it below $1-\epsilon$ brings no further gain. The flat regions limit the incentive to keep moving in the favored direction; they are not hard constraints on probabilities or guarantees of bounded policy divergence [1].

Why include a reference policy?

Clipping compares the current policy with the rollout snapshot. The KL penalty instead discourages drift from a longer-lived reference policy. Its strength is set by $\beta \geq 0$. The original GRPO formulation uses the sampled penalty

$$\widehat{d}^{\text{KL}}_{i,t} = x_{i,t} - \log x_{i,t} - 1 \geq 0, \qquad x_{i,t} = \frac{\pi_{\text{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}. \tag{6}$$

At a fixed prefix, its expectation equals $D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$ when the token is sampled from π_θ. Reused old-policy samples do not generally retain that equality without a sampling correction. A separate KL term may still update the policy when a group's reward advantages are all zero.

3 Two variants: DAPO and GSPO

3.1 DAPO: keep useful groups and handle long responses

DAPO [5] combines asymmetric clipping with a token-averaged objective:

$$C_{\ell,h}(\rho, A) = \min\{\rho A, \mathrm{clip}(\rho, 1-\epsilon_\ell, 1+\epsilon_h)A\}, \tag{7}$$

$$J_{\mathrm{DAPO}} = \mathbb{E}_{\text{retained groups}}\left[\frac{\sum_{i=1}^{G}\sum_{t=1}^{T_i} C_{\ell,h}(\rho_{i,t}, \widehat{A}_i)}{\sum_{i=1}^{G} T_i}\right]. \tag{8}$$

Clip-Higher sets $\epsilon_h > \epsilon_\ell$, allowing more room for probability increases. **Token averaging** gives equal normalization weight to each token: responses of lengths 20 and 80 receive total weights 1/5 and 4/5, versus GRPO's 1/2 each. The reported recipe omits reference KL.

Dynamic sampling (additional rollouts). A rollout is one generated answer. DAPO keeps a prompt's group only when some answers are correct and some are wrong. All-correct and all-wrong groups provide no relative signal under a correctness-only reward. It samples more prompts and their groups until enough usable groups fill the training batch. For example, if an update needs 64 groups and only 40 survive filtering, generation continues until 64 are available. The group size G can stay fixed; the amount of generation per update is dynamic. Wrong answers within a retained group are kept for comparison.

Overlong reward shaping. A response cut off at the token limit may contain valid but unfinished reasoning. Automatically treating it as a failed solution can introduce misleading feedback. The paper first tests masking truncated responses out of the loss. It also introduces a soft length penalty, added to the correctness reward before computing advantages:

$$R_{\mathrm{length}}(T) = -\,\mathrm{clip}\left(\frac{T-(L_{\max}-B)}{B},\, 0,\, 1\right), \qquad R = R_{\mathrm{correct}} + R_{\mathrm{length}}. \tag{9}$$

Here $L_{\max}$ is the length limit and $B > 0$ the penalty buffer. There is no penalty before the buffer; it then increases linearly to -1 at the limit. For an illustrative $L_{\max} = 8000$ and $B = 2000$, lengths 6000, 7000, and 8000 receive penalties 0, -0.5, and -1. This encourages finishing within the budget without penalizing every long answer equally.

3.2 GSPO: one clipping decision per response

GSPO [6] replaces individual token ratios with their geometric mean:

$$s_i(\theta) = \left(\frac{\pi_\theta(o_i \mid q)}{\pi_{\mathrm{old}}(o_i \mid q)}\right)^{1/T_i} = \exp\left(\frac{1}{T_i}\sum_{t=1}^{T_i}\log\rho_{i,t}\right), \tag{10}$$

$$J_{\mathrm{GSPO}} = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} C_\epsilon(s_i, \widehat{A}_i)\right]. \tag{11}$$

For example, token ratios $(2, 0.5, 1, 1)$ mean that one sampled token became twice as probable, another half as probable, and two stayed unchanged. GSPO combines them into $s_i = (2 \cdot 0.5 \cdot 1 \cdot 1)^{1/4} = 1$ and applies clipping once to the response. GRPO applies clipping separately to each token ratio. A ratio of one can still have a nonzero gradient: it does not imply no learning.

The displayed GSPO objective omits KL. The T_i-th root controls scale, so s_i is a normalized surrogate rather than the raw sequence importance weight. Its clipping thresholds therefore need their own tuning.

4 Reward design: what should the group compare?

Rewards can come from executable tests, mathematical verifiers, learned scalar scorers, or LLM judges. A generative reward model produces judgments that are converted into scores; it is distinct from the critic that predicts future return.

Relative does not mean rank-only

Group standardization removes a common offset and positive scale: replacing R_i by aR_i+b, with $a > 0$, leaves the advantages unchanged when $\delta = 0$ and $\sigma_R > 0$ (approximately so with a small stabilizer). Nonlinear changes can alter score gaps even if the ranking stays fixed. For example, $(0, 1, 2)$ and $(0, 1, 100)$ have the same ordering, but the middle answer's advantage is respectively 0 and about -0.696.

Design A: score explicit criteria

A transparent rubric combines binary checks with nonnegative weights:

$$R_i = \frac{\sum_{k=1}^{K} w_k c_k(q, o_i)}{\sum_{k=1}^{K} w_k}, \qquad c_k \in \{0, 1\}, \quad w_k \geq 0, \quad \sum_k w_k > 0. \tag{12}$$

Criteria might check correctness, supporting evidence, and the requested format. Equal weights give the fraction of criteria met. Choose the tradeoffs carefully: if formatting can compensate for a wrong answer, the model may increase reward without becoming more useful. A correctness gate can prevent that compensation when correctness is essential.

Design B: convert pairwise preferences into rewards

Compare answers in pairs, then score wins and ties:

$$R_i^{\text{pair}} = \frac{1}{G-1} \sum_{j \neq i} \left[\mathbf{1}(o_i \succ o_j) + \tfrac{1}{2} \mathbf{1}(o_i \sim o_j) \right]. \tag{13}$$

Here $\succ$ means "judged better" and $\sim$ means a tie. Judge each unordered pair once and assign complementary outcomes. A strict ordering of four answers gives scores $0, 1/3, 2/3, 1$, which can then be standardized within the group.

All-pairs judging costs $G(G-1)/2$ comparisons: 120 at $G = 16$. LLM preferences can be inconsistent or position-sensitive, so randomize answer order, check some pairs in reverse, and allow ties. Pairwise judging may simplify evaluation, but it does not automatically make it more reliable. Validate the resulting policy on held-out prompts, not only on its training reward.

References

[1] J. Schulman et al. (2017). *Proximal Policy Optimization Algorithms*. arXiv:1707.06347.

[2] L. Ouyang et al. (2022). *Training language models to follow instructions with human feedback*. arXiv:2203.02155.

[3] Z. Shao et al. (2024). *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. Section 4.1. arXiv:2402.03300.

[4] DeepSeek-AI et al. (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv:2501.12948.

[5] Q. Yu et al. (2025). *DAPO: An Open-Source LLM Reinforcement Learning System at Scale*. Sections 2–3. arXiv:2503.14476.

[6] C. Zheng et al. (2025). *Group Sequence Policy Optimization*. Section 4. arXiv:2507.18071.