Multi-Agent Roles as Reward Channels: A Diagnostic Study of Verifier-Style Rewards for Mathematical Reasoning

Anonymous authors
Paper under double-blind review

September 26, 2026

Abstract

Multi-agent systems for mathematical reasoning typically run a solver, a verifier and a critic as separate models, which multiplies inference cost. We study a cheaper alternative in which these roles are collapsed into separate reward channels evaluated on a single policy's own generations: a correctness channel, a verifier channel that re-extracts and normalises the model's answer through the solver's own extraction pipeline, and a monitor channel that classifies each generation as valid, refusal, invalid format, no valid extraction, or repetition. The same monitor categories serve as an evaluation metric, so the training signal and the reported reliability metric are defined identically.

Applying this design to DeepSeek-Math-7B-Instruct on olympiad-level problems, we find that neither supervised fine-tuning on chain-of-thought traces nor subsequent GRPO improves accuracy over the untrained model on our test set: pass@1 is 8.5% (95% CI [5.4, 13.2]) for the base model, 7.0% [4.2, 11.4] after GRPO alone, 4.5% [2.4, 8.3] after supervised fine-tuning, and 2.0% [0.8, 5.0] after both ($n = 200$ problems). Training diagnostics identify a mechanism specific to this design. Because the generation limit used during training (512 tokens) is far below the length of the chain-of-thought traces used for supervision, post-SFT completions reach the limit on every logged step (mean 511.6 tokens, against 403.5 for the untrained model) and rarely emit a final answer. The monitor channel is then pinned at its minimum value for every sample ($r_{\mathrm{fmt}} = -0.2000$, sd 0.0000), and a reward channel with no within-group variance cancels in the group-relative advantage and contributes nothing to the gradient. Total reward variance falls from 0.204 to 0.124 and the policy moves very little (KL ≈ 0.003 over 58 steps). We argue that channel saturation is the characteristic failure mode of collapsing agent roles into a reward vector: in a system of separate agents a saturated verifier is observable, whereas here it disappears silently. We recommend monitoring per-channel within-group variance as a standard diagnostic, and report behavioural metrics that localise where degradation occurs when accuracy alone cannot.

1 Introduction

Large language models now solve a substantial fraction of competition-level mathematics problems by producing explicit chains of reasoning (Wei et al., 2022; Shao et al., 2024; DeepSeek-AI, 2025). A central difficulty in training them is that a long derivation can be judged only by its final answer, which provides a single sparse signal for many tokens of reasoning. Two families of methods address this. Verifier-based approaches train an additional model to judge solutions, either at the outcome level (Cobbe et al., 2021) or step by step (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024). Multi-agent approaches go further and run several models that debate, critique or refine one another's outputs at inference time (Du et al., 2024; Liang et al., 2024; Madaan et al., 2023; Shinn et al., 2023). Both buy signal with parameters: every additional role is a model to train, to serve, or both, and Smit et al. (2024) find that the gains do not always survive being compared at equal inference cost.

Reinforcement learning with verifiable rewards offers a way to use a checkable signal without a learned judge. Group Relative Policy Optimization (GRPO; Shao et al., 2024) is now widely adopted for this purpose (DeepSeek-AI, 2025): it samples a group of completions per problem and normalises each reward against the group mean and standard deviation, removing the critic network entirely. Subsequent work has examined when this signal is weak or absent. Yu et al. (2025) show that groups in which every sample receives the same reward contribute no gradient and filter

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

them out; Le et al. (2026) and Liu et al. (2025a) recover learning signal from such zero-variance prompts rather than discarding them; and Yue et al. (2025) argue that RLVR largely reweights solutions the base model can already produce.

In this work we study a construction that sits between these lines: the roles of a multi-agent system, expressed not as agents but as separate *reward channels* over a single policy's own completions. A solver channel compares the extracted answer to the reference; a verifier channel re-extracts and normalises the model's answer through the same pipeline used at inference and scores agreement; a monitor channel assigns each completion to one of six behavioural categories and converts that label into a reward. Each channel asks what its corresponding agent would ask, but is computed by deterministic code from the completion itself, so no second model is trained or served. Figure 1 contrasts the two designs.

This collapse of roles into channels admits a failure mode that the multi-agent form does not. Because GRPO normalises within a group, a channel whose value is identical across every sample in that group cancels exactly in the advantage and contributes nothing to the gradient, however informative it might be in principle. We call this *channel saturation*. It is not the zero-variance condition studied by Yu et al. (2025) and Le et al. (2026), which concerns the aggregate reward of a prompt and is therefore detectable by existing filters: in our runs the aggregate reward retains variance while a single channel inside it is constant, so no aggregate-level check fires. The difference matters because a system of separate agents makes the same condition visible – a verifier that returns one verdict on every query is apparent in its outputs – whereas a saturated channel appears in the training logs only as a constant offset, easily mistaken for a strict penalty.

We report a case in which this occurs. Fine-tuning DeepSeek-Math-7B-Instruct on chain-of-thought traces from OpenMathReasoning (Moshkov et al., 2025) under a 512-token generation limit produced a policy whose completions reach that limit on every logged training step, so a final answer is rarely emitted. The monitor channel was then pinned at its minimum value for every sample, and the policy moved very little under GRPO. Accuracy on our olympiad test set decreased monotonically across the pipeline rather than improving.

Our contribution is a failure mode of decomposed rewards under group-relative optimisation, together with the instrumentation that detects it. Multi-component reward shaping is standard practice, and expressing the components as agent roles is a framing rather than a method; what has not been noted, to our knowledge, is that a component can be entirely inert while every aggregate-level check reports a healthy run. We make this precise in Section 4, showing that channel saturation follows directly from the group-relative advantage and that dependence between channels, a tight generation budget and a small group size each make it more likely. We then report measurements of it occurring (Section 6.3): after fine-tuning, completions sit at the generation limit on 100% of logged steps and the format component of the monitor channel is constant at its floor (-0.2000, $\mathrm{sd} = 0.0000$), against 403.5 tokens and a varying format reward for the untrained policy. Because that value is the channel's minimum, every completion in those batches took the same branch, so within-group variance was zero throughout. Finally, we pair this with a behavioural taxonomy used identically as training signal and evaluation metric (Section 6.2), which localises degradation that accuracy alone does not explain – fine-tuning introduced an answer-extraction failure mode in 0.56–0.62% of samples that is absent from the untrained model - and we evaluate four conditions (base, GRPO only, SFT, SFT + GRPO) on 200 olympiad problems with 95% Wilson intervals throughout.

We do not claim that the reward-channel construction improves accuracy; on our evaluation it does not, and our RL configuration is too small to support conclusions about GRPO in general (Section 8). What we offer is a mechanism that explains the outcome, measurements that expose it, and a diagnostic we recommend for any group-relative method with a decomposed reward: log the within-group standard deviation of each channel separately, and treat a channel with zero variance as inactive rather than as a strong penalty.

2 Related work

Verifiers as separate models. Cobbe et al. (2021) train an outcome verifier to rerank sampled solutions, and Lightman et al. (2024) show that process reward models, supervised on step-level labels, outperform outcome supervision on competition mathematics. Uesato et al. (2022) compare the two forms of feedback directly, and Wang et al. (2024) obtain step labels automatically from rollouts rather than from annotators. These methods share the assumption we relax: the verifier is a distinct model with its own parameters and its own inference cost. Our verifier channel is not a model at all; it is a function of the solver's output computed with the solver's own extraction code. This makes it far cheaper and also far weaker, since it can only check answer-level agreement and formal well-formedness, not the validity of intermediate steps. Where step-level supervision is available, the methods above provide a denser signal than

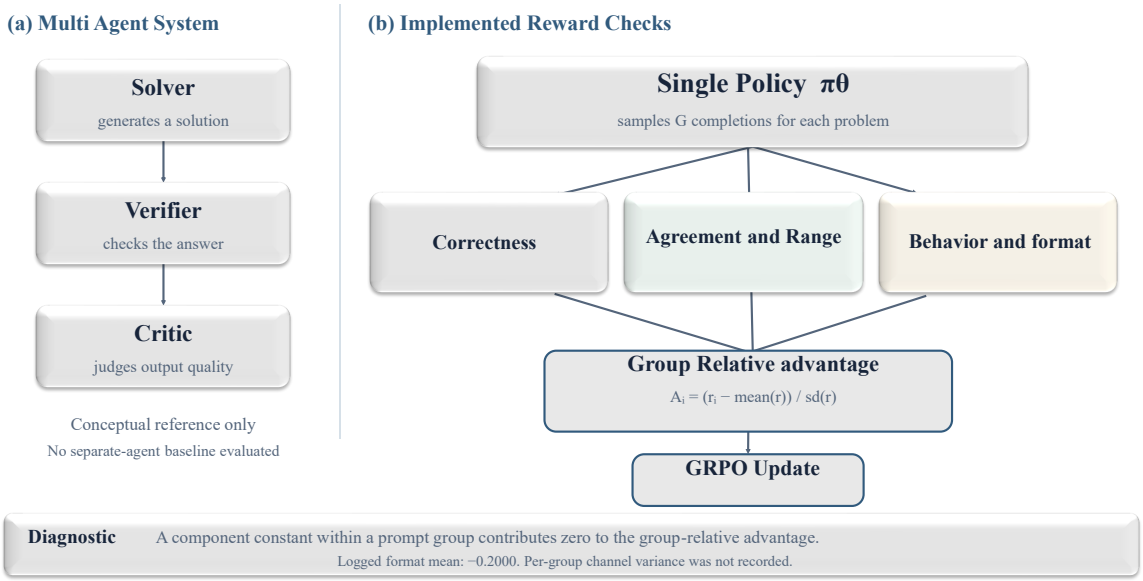

Figure 1: Agent roles as reward channels. (a) A conventional multi-agent system serves solver, verifier and critic as separate models; a saturated agent is observable because it returns the same verdict on every query. (b) We score a single policy's completions with three reward channels that reuse its own answer-extraction pipeline, so no additional model is trained or served. (c) The failure this admits: when the generation limit truncates completions before an answer is emitted, the monitor channel takes the same value for every sample in a group, cancels in the group-relative advantage, and stops contributing to the gradient without appearing as an error. Numbers are measured values from Table 4.

ours.

Multiple agents that coordinate at inference time. Debate (Du et al., 2024; Liang et al., 2024) and iterative self-critique (Madaan et al., 2023; Shinn et al., 2023) run several models, or several passes of one model, and aggregate their outputs at inference time. Communication is prompted rather than learned, and the aggregation affects only the answer returned, not the weights. Smit et al. (2024) find that debate does not reliably outperform self-consistency once inference cost is held constant. Our setting differs in both respects: the roles interact only through the reward used in training, and at inference time a single policy runs once. Huang et al. (2024) report that intrinsic self-correction without external feedback does not improve, and can degrade, reasoning accuracy; our monitor channel is correspondingly restricted to checks that do not require the model to judge its own correctness.

Verification signals inside a single model. Kumar et al. (2025) train a single model to correct its own answers with multi-turn RL, and Zhang et al. (2025) train verifiers as generators so that verification and generation share an architecture. Weng et al. (2023) use backward verification by the same model to rerank candidates. Our construction is simpler and more restrictive than these: we do not train a verification behaviour, we only shape the reward with quantities that can be computed deterministically from a completion. The comparison we care about is therefore not which method reasons better, but what is lost when verification is reduced to a reward term, which is the subject of Section 6.3.

GRPO and when it fails. GRPO (Shao et al., 2024) replaces the learned critic with a group-relative baseline, and DeepSeek-AI (2025) scale it with rule-based rewards. Subsequent analyses identify conditions under which it provides little signal: Yu et al. (2025) show that groups in which all samples receive the same outcome contribute no gradient and propose dynamic sampling to avoid them, Liu et al. (2025b) identify normalisation biases in the objective, and Yue et al.

(2025) argue that RLVR mainly reweights solutions the base model can already produce. Our observation differs from this line of work operationally rather than conceptually. DAPO's filter, and the detection step in Le et al. (2026) and Liu et al. (2025a), all test whether the *aggregate* group reward has zero variance. In our SFT + GRPO run the aggregate within-group standard deviation is 0.124 (Table 4), so none of these checks would fire, while one of five channels is constant throughout training. Reward decomposition therefore creates a blind spot in existing instrumentation, rather than supplying another instance of a condition those methods already handle.

Fine-tuning that degrades a specialised model. That post-training can reduce capability is established: Ouyang et al. (2022) name the alignment tax, Kumar et al. (2022) show that fine-tuning distorts pretrained features, and Gudibande et al. (2024) find that imitating a stronger model's outputs transfers style rather than capability. Closest to our setting, Li et al. (2025) report that small models often fail to benefit from long chain-of-thought distillation. Our result is consistent with this line of work and adds a specific, measurable mechanism for one instance of it: the supervision traces are far longer than the generation limit used in training and evaluation, so the model is optimised on truncated reasoning.

Evaluation beyond accuracy. Holtzman et al. (2020) characterise degenerate generation, and Liang et al. (2023) argue for multi-metric evaluation. Mirzadeh et al. (2025) show that mathematical accuracy is sensitive to perturbations that leave the problem unchanged, which motivates reporting interval estimates rather than point estimates. Our behavioural taxonomy is a small instance of this: six mutually exclusive categories, computed by the same code used for the monitor reward channel, so that the training signal and the evaluation metric cannot drift apart.

3 Preliminaries

Notation. We define an autoregressive language model parameterised by θ as a policy π_θ. Let q denote a problem, $a^\star$ its reference answer, and $\mathcal{D}$ the problem set. For a completion o generated under a maximum length $L_{\max}$, its likelihood is

$$\pi_\theta(o \mid q) = \prod_{t=1}^{|o|} \pi_\theta\big(o_t \mid q, o_{<t}\big), \tag{1}$$

where $|o| \le L_{\max}$ is the number of generated tokens. We write $e(\cdot)$ for answer extraction and $\nu(\cdot)$ for normalisation, both taken unchanged from the inference pipeline; $e(o) = \varnothing$ denotes a completion from which no answer can be extracted.

Group Relative Policy Optimization. GRPO (Shao et al., 2024) estimates the baseline from group-level rewards rather than from a learned critic. For a problem q, the behaviour policy $\pi_{\theta_{\text{old}}}$ samples a group of G completions $\{o_i\}_{i=1}^{G}$, and the advantage of the i-th completion is obtained by normalising the group rewards $\{r_i\}_{i=1}^{G}$:

$$\hat{A}_i \;=\; \frac{r_i - \text{mean}\big(\{r_j\}_{j=1}^{G}\big)}{\text{std}\big(\{r_j\}_{j=1}^{G}\big)}. \tag{2}$$

The policy is then optimised with a clipped objective and a KL penalty towards a fixed reference policy π_{ref}:

$$\mathcal{J}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\,\{o_i\}\sim\pi_{\theta_{\text{old}}}}\left[\frac{1}{G}\sum_{i=1}^{G}\Big(\min\big(\rho_i(\theta)\hat{A}_i,\ \text{clip}(\rho_i(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_i\big) - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\Big)\right], \tag{3}$$

where $\rho_i(\theta)$ is the importance ratio between the new and old policy. The property we rely on throughout is visible in Equation (2): because the numerator is a deviation from the group mean, *only differences within a group affect the update*.

4 Methodology

Our construction expresses the solver, verifier and monitor roles of a multi-agent reasoning system as additive reward channels over a single policy's own completions, so that no second model is trained or served. Section 4.1 defines

the five channels and the role each one replaces. Section 4.2 shows that this construction admits a failure mode the multi-agent form does not: a channel whose value is constant within a group cancels in Equation (2) and leaves the gradient without appearing as an error. It further explains why truncation and small group sizes make that condition likely. Section 4.3 states the resulting procedure and the per-channel diagnostic we recommend alongside it.

4.1 Roles as reward channels

We write the scalar reward as a sum of channels,

$$r_i \;=\; \underbrace{r_{\mathrm{corr}}(o_i, a^\star)}_{\text{solver}} \;+\; \underbrace{r_{\mathrm{ver}}(o_i, a^\star) + r_{\mathrm{rng}}(o_i)}_{\text{verifier}} \;+\; \underbrace{r_{\mathrm{beh}}(o_i) + r_{\mathrm{fmt}}(o_i)}_{\text{monitor}}, \tag{4}$$

each computed from o_i by deterministic code.

Solver channel. r_{corr} is the outcome signal and the dominant term:

$$r_{\mathrm{corr}}(o_i, a^\star) = \begin{cases} +1.0, & \nu(e(o_i)) = \nu(a^\star), \\ -1.0, & \nu(e(o_i)) \neq \nu(a^\star), \\ \;\;0.0, & e(o_i) = \varnothing. \end{cases} \tag{5}$$

The third branch matters later: a completion from which no answer can be extracted is neither rewarded nor penalised.

Verifier channel. r_{ver} re-derives the answer through the same extraction and normalisation functions used at inference and scores agreement,

$$r_{\mathrm{ver}}(o_i, a^\star) = \begin{cases} +0.15, & \nu(e(o_i)) = \nu(a^\star), \\ -0.15, & \text{otherwise, including } e(o_i) = \varnothing, \end{cases} \tag{6}$$

with a companion range check $r_{\mathrm{rng}} \in \{+0.1, -0.1\}$ according to whether the extracted value lies in the expected numeric range. This asks what a verifier agent would ask — does the stated answer survive independent re-extraction and comparison? — but reuses the solver's own pipeline, so the check is cheap and *not* independent of the solver's formatting behaviour. That dependence is what makes it vulnerable to the saturation below.

Monitor channel. A classifier $c(\cdot)$ assigns each completion to exactly one of six categories,

$$c(o_i) \in \{\texttt{valid}, \texttt{no_output}, \texttt{refusal}, \texttt{invalid_format}, \texttt{no_valid_extraction}, \texttt{repetition}\}, \tag{7}$$

by checking in order for an empty completion, refusal phrases, a final-answer marker without an extractable value, and repetition. The behaviour channel gives $r_{\mathrm{beh}} = +0.15$ for `valid`, -0.15 for the degenerate categories and 0 otherwise, while the format channel is binary and admits no neutral branch:

$$r_{\mathrm{fmt}}(o_i) = \begin{cases} +0.2, & o_i \text{ is well formed and } e(o_i) \neq \varnothing, \\ -0.2, & \text{otherwise.} \end{cases} \tag{8}$$

The same $c(\cdot)$ is used unchanged at evaluation time (Section 6.2), so the training signal and the reported reliability metric are one implementation rather than two. Table 1 summarises the five channels.

4.2 Channel saturation

Write $r_i^{(k)}$ for the value of channel k on completion o_i, so that $r_i = \sum_k r_i^{(k)}$. Substituting into Equation (2) gives

$$\hat{A}_i = \frac{1}{\mathrm{std}(\{r_j\})} \sum_k \Big(r_i^{(k)} - \mathrm{mean}_j\big(r_j^{(k)}\big) \Big), \tag{9}$$

from which the failure mode follows immediately.

Table 1: Reward channels, the multi-agent role each replaces, and its range. Only r_{corr} and r_{beh} have a neutral branch; r_{ver}, r_{rng} and r_{fmt} are binary and therefore take an extreme value on every completion. Three of the five depend on whether an answer can be extracted.

Channel	Role	Values	Neutral branch	Needs extraction
r_{corr}	solver	$+1.0$ / -1.0 / 0.0	yes	yes
r_{ver}	verifier	$+0.15$ / -0.15	no	yes
r_{rng}	verifier	$+0.1$ / -0.1	no	yes
r_{beh}	monitor	$+0.15$ / -0.15 / 0.0	yes	no
r_{fmt}	monitor	$+0.2$ / -0.2	no	yes

If channel k takes the same value on every completion in a group, that is $r_1^{(k)} = \cdots = r_G^{(k)}$, then $r_i^{(k)} - \text{mean}_j(r_j^{(k)}) = 0$ for every i, so the channel contributes nothing to any $\hat{A}_i$ and has no influence on the update for that group, irrespective of the magnitude of its value. We refer to this condition as *channel saturation*. The algebra is immediate; what is not immediate is that the condition arises in practice, and that it leaves no trace in the quantities practitioners normally monitor.

The consequence is not that a saturated channel contributes weakly; it contributes nothing. Its only visible effect is a constant offset in the aggregate reward, which is easily mistaken for a channel that is simply pessimistic. Three properties of the construction make this condition more likely than it may appear.

(i) The channels are not independent. By Table 1, r_{corr}, r_{ver}, r_{rng} and r_{fmt} all depend on whether an answer can be extracted, so a single condition that prevents extraction affects four of the five channels at once.

(ii) Truncation is such a condition. If $L_{\max}$ is shorter than the length at which the policy emits its final answer, then $e(o_i) = \varnothing$ for *every* sample in the group. Since r_{fmt} is binary (Equation (8)), saturation is total rather than partial: all G completions take the -0.2 branch and no intermediate value is available. At the same time, by Equation (5) the dominant correctness channel returns 0.0, so the strongest term in the reward falls silent as well. This produces an incentive we did not intend: under a tight generation budget, not finishing ($r_{\text{corr}} = 0.0$) is scored better than finishing incorrectly ($r_{\text{corr}} = -1.0$).

(iii) Small groups saturate easily. A channel is inactive for a problem whenever its G samples happen to agree, which is far more likely for small G. Our $G = 2$ is therefore a worst case; group sizes of 8–64 are typical in recent GRPO work. Note that larger G reduces the frequency of saturated groups but does not remove the mechanism, since under truncation all samples fail identically regardless of how many are drawn.

A system of separate agents does not hide this condition: a verifier that returns the same verdict on every query is visible in its outputs. Collapsing the roles into reward channels removes that observability, and this is the cost that accompanies the saving in inference.

4.3 Procedure and diagnostic

Algorithm 1 states the training loop. The only addition to standard GRPO is steps 10–13: we log the within-group standard deviation of each channel separately and record the fraction of groups in which it is zero. A channel with $\sigma^{(k)} = 0$ in most groups is inactive whatever its mean, and should be treated as a defect in the reward design rather than as a strong penalty. This is the per-channel analogue of the all-correct/all-incorrect group filtering that Yu et al. (2025) apply to the aggregate reward, and it detects a condition that aggregate-level checks do not: in our runs the total reward retains variance (Table 4) while one channel inside it is constant.

5 Experimental setup

Model and data. All experiments use DeepSeek-Math-7B-Instruct (Shao et al., 2024) as the base model. Supervised fine-tuning data comes from the chain-of-thought subset of OpenMathReasoning (Moshkov et al., 2025). We retain problems that satisfy three conditions: the extracted answer field is populated, the final answer is a single non-negative integer, and that integer lies in $[0,\ 999{,}999]$. We additionally filter on the dataset's reasoning-quality rating, keeping ratings in $[1, 5]$, and on length, admitting inputs of at most 4,096 and targets of at most 1,024 tokens. From the problems

Algorithm 1 Role-structured reward channels with per-channel saturation diagnostic

Require: policy π_θ; problems $\mathcal{D} = \{(q_n, a_n^\star)\}_{n=1}^{N}$; channels $\{r^{(k)}\}_{k=1}^{K}$ (Equation (4)); group size G; generation limit $L_{\max}$; KL coefficient β; steps T
Ensure: updated policy π_θ; saturation rates $\{s^{(k)}\}_{k=1}^{K}$

$z^{(k)} \leftarrow 0$ for $k = 1, \ldots, K$; $\quad m \leftarrow 0$ — *counters*
for $t = 1, \ldots, T$ **do**
  sample a mini-batch $\mathcal{B} \subseteq \mathcal{D}$
  for each $(q, a^\star) \in \mathcal{B}$ **do**
    sample $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$, $|o_i| \le L_{\max}$
    for $k = 1, \ldots, K$ **do**
      $r_i^{(k)} \leftarrow r^{(k)}(o_i, a^\star)$, $i = 1, \ldots, G$ — *every channel scores every completion*
    end for
    $r_i \leftarrow \sum_k r_i^{(k)}$; $\quad \hat{A}_i \leftarrow \dfrac{r_i - \text{mean}(\{r_j\})}{\text{std}(\{r_j\})}$
    for $k = 1, \ldots, K$ **do** — *saturation diagnostic*
      $\sigma^{(k)} \leftarrow \text{std}\big(\{r_i^{(k)}\}_{i=1}^{G}\big)$
      $z^{(k)} \leftarrow z^{(k)} + \mathbf{1}\big[\sigma^{(k)} = 0\big]$ — 1 *when the channel is inert here*
    end for
    $m \leftarrow m + 1$
  end for
  update π_θ by maximising Equation (3) on $\{(q, o_i, \hat{A}_i)\}$
end for
$s^{(k)} \leftarrow z^{(k)}/m$ for $k = 1, \ldots, K$ — *report with each channel's mean*
return π_θ, $\{s^{(k)}\}$

that remain we sample $N = 50{,}000$ with seed 42. The restriction to integer answers in a bounded range is what makes the reward channels of Section 4 computable by string comparison rather than by a learned model.

Each example is formatted as a single prompt-completion pair with an explicit answer delimiter:

Segment	Content
Prompt	`Problem:` {problem}
Completion	`Solution:` {reasoning}
	`FINAL_ANSWER:` {answer}

The `FINAL_ANSWER` delimiter is the anchor for the extraction function $e(\cdot)$ used both at inference and inside every reward channel, so a completion that never reaches this delimiter is unparseable by construction. This is the structural reason truncation propagates into the reward, as analysed in Section 4.2.

We split the 50,000 problems 70/10/10/10 into train, validation, mining and test sets, stratifying on unique problem text so that no problem appears in more than one split. This matters because OpenMathReasoning contains multiple reasoning traces per problem; splitting on traces rather than problems would leak test problems into training. The resulting partition contains 35,021 training, 5,018 validation, 4,949 mining and 5,012 test problems. The mining split is held out so that the GRPO stages train on prompts the policy has not been evaluated on.

Pipeline. We compare four conditions: **Base**, the released instruct model with no further training; **SFT**, one epoch of supervised fine-tuning on the training split; **GRPO only**, GRPO applied directly to the base model; and **SFT + GRPO**, GRPO applied to the fine-tuned model. Each GRPO stage trains on at most 250 prompts drawn from the mining split, selected where the corresponding policy failed.

Training configuration. SFT uses LoRA (Hu et al., 2022) with $r = 16$, $\alpha = 32$, applied to all attention and MLP projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`), trained for one epoch at learning rate 1.80×10^{-5} in bf16 with gradient checkpointing, for a planned 4,377 optimiser steps with evaluation

every 500 steps. Training reached a wall-clock limit after approximately six hours and was resumed from the step-4000 checkpoint to completion.

GRPO uses the same LoRA configuration with AdamW at learning rate 5.0×10^{-6}, per-device batch size 2, gradient accumulation 4, and KL coefficient $\beta = 0.04$. It ran for 58 steps (SFT + GRPO) and 56 steps (GRPO only), approximately one epoch over the mined prompts in each case. Prompts are truncated to 384 tokens.

Two settings are central to the analysis. First, the group size is $G = 2$: each problem contributes two sampled completions, so the advantage in Equation (2) compares a single pair. This is small relative to the group sizes of 8–64 used in recent GRPO work, and it makes zero within-group variance more likely, since only two samples need to coincide. Second, *both training and evaluation cap generation at* $L_{\max} = 512$ *tokens*, while the data filter admits target solutions of up to 1,024 tokens; the model is therefore supervised on sequences up to twice the length it is permitted to generate. Reward magnitudes per channel are given in Table 1.

Evaluation. We evaluate on 200 problems drawn from the test split, reporting pass@1 from a single sample and maj@8, a majority vote over eight samples drawn at temperature 0.7 and top-$p = 0.95$, both under the same 512-token generation limit used in training. Behavioural reliability is computed over all $200 \times 8 = 1{,}600$ sampled completions per condition using the classifier $c(\cdot)$ of Section 4. We report 95% Wilson score intervals for every proportion. Because our evaluation code retained aggregate rates rather than per-problem outcomes, we cannot report paired tests between conditions; see Section 8.

Compute. All runs used single NVIDIA A100 80 GB nodes. Supervised fine-tuning required 9.9 GPU-hours across the interrupted and resumed runs, with 8 CPU cores and 96 GB of host memory; failure mining took 2.2 hours under the same allocation. Each GRPO run used 16 CPU cores and 192 GB of host memory, taking 2.6 hours (SFT + GRPO) and 2.3 hours (GRPO only). Evaluating the four conditions took 1.5, 3.1, 3.5 and 3.7 hours respectively. The reported pipeline totals approximately 28.8 GPU-hours across successful runs, excluding failed and cancelled attempts.

Our GRPO runs are short by the standards of recent work on reinforcement learning with verifiable rewards: 56–58 optimiser steps, against the hundreds to thousands used in larger studies. This limits what we can conclude about GRPO's potential on this task. It does not affect the observation in Section 6.3, which concerns the reward decomposition rather than the amount of training: a channel with zero within-group variance contributes nothing regardless of how many steps are run.

6 Results

We organise the results around three questions. Does the pipeline improve accuracy (Section 6.1)? Where does any degradation appear in the model's behaviour (Section 6.2)? And what do the training signals show (Section 6.3)?

6.1 Accuracy

Table 2 and Figure 2 report accuracy for the four conditions. Neither stage improves on the untrained model. The base model solves 17 of 200 problems at pass@1 (8.5%, 95% CI [5.4, 13.2]); after supervised fine-tuning this falls to 9 of 200 (4.5% [2.4, 8.3]), and after the subsequent GRPO stage to 4 of 200 (2.0% [0.8, 5.0]). The same ordering holds for maj@8, where the base model solves 24 of 200 (12.0% [8.2, 17.2]) against 11 of 200 (5.5% [3.1, 9.6]) after fine-tuning.

Two comparisons deserve different treatment. The interval for SFT + GRPO at pass@1 ([0.8, 5.0]) does not overlap that of the base model ([5.4, 13.2]), so the degradation across the full pipeline is unlikely to be a sampling artefact. By contrast, GRPO applied to the base model changes pass@1 from 8.5% to 7.0% with almost completely overlapping intervals; we treat this as no detectable effect rather than as a small degradation. At $n = 200$, differences smaller than roughly five percentage points are not resolvable, and we do not interpret them.

6.2 Where the degradation appears

Accuracy alone does not indicate whether the model produces worse reasoning or merely unusable output. Table 3 applies the monitor classifier to all 1,600 sampled completions per condition. Both untrained conditions produce a well-formed, extractable answer in 100.00% of samples. After supervised fine-tuning this falls to 98.75%, and after

Table 2: Accuracy on 200 olympiad test problems, with 95% Wilson intervals. maj@8 is a majority vote over eight samples at temperature 0.7. The final column gives the raw counts. Neither training stage improves on the untrained model.

Model	pass@1 (%)	95% CI	maj@8 (%)	95% CI	solved (p@1 / maj@8)
Base (no training)	8.5	[5.4, 13.2]	12.0	[8.2, 17.2]	17/200 vs 24/200
GRPO only	7.0	[4.2, 11.4]	11.0	[7.4, 16.1]	14/200 vs 22/200
SFT	4.5	[2.4, 8.3]	5.5	[3.1, 9.6]	9/200 vs 11/200
SFT + GRPO	2.0	[0.8, 5.0]	6.5	[3.8, 10.8]	4/200 vs 13/200

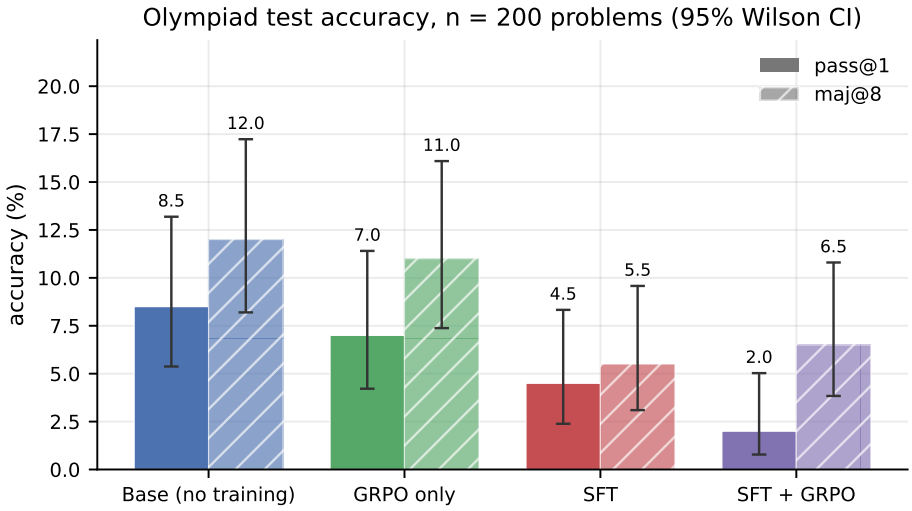

Figure 2: Accuracy across the four conditions. Error bars are 95% Wilson intervals. The intervals for base and GRPO-only overlap almost entirely, whereas the interval for SFT + GRPO at pass@1 is disjoint from that of the base model.

GRPO to 98.31%. The lost mass appears in two categories that are absent from the base model: refusals (0.69% and 1.06%) and completions in which a final-answer marker is present but no value can be extracted (0.56% and 0.62%).

The size of this effect is worth stating precisely, because it is easy to overread. A 1.25–1.69 percentage-point loss of valid outputs cannot account for a 4–6.5 point loss of accuracy; most of the degradation is wrong answers, not unusable ones. What the taxonomy does show is that fine-tuning introduced a *new* failure mode rather than simply making the model less accurate, and the nature of that mode — generations that announce an answer they never produce — points towards truncation, which Section 6.3 confirms from the training side.

6.3 Training diagnostics: a saturated channel

Table 4 and Figure 4 report per-channel statistics from the two GRPO runs, and they differ in a way that identifies the mechanism.

For the fine-tuned policy, mean completion length is 511.6 tokens against a limit of 512, and every logged step sits at the limit. The untrained policy averages 403.5 tokens and never reaches it. That is, supervised fine-tuning on long chain-of-thought traces produced a model that exhausts its generation budget before finishing, which is consistent with the traces themselves being much longer than the budget used in training (Moshkov et al., 2025).

The consequence for the reward decomposition is direct. For the fine-tuned policy the format component of the monitor channel takes the value -0.2000 with standard deviation 0.0000 across logged steps. This licenses a stronger statement than the batch-level logging might suggest. Because -0.2 is the minimum value r_{fmt} can take (Equation (8)), a batch mean of exactly -0.2000 is attainable only if *every* sampled completion in that batch took the negative branch. Within-group variance is therefore zero for every group contributing to those steps, and the argument of Section 4.2

Table 3: Behavioural reliability over 1,600 sampled completions per condition, computed with the same classifier used for the monitor reward channel. The `no_valid_extraction` category is absent from both untrained conditions and appears only after supervised fine-tuning.

Model	Valid (%)	Refusal (%)	Invalid format (%)	No extraction (%)	Repetition (%)	No output (%)	Samples
Base (no training)	100.00	0.00	0.00	0.00	0.00	0.00	1600
GRPO only	100.00	0.00	0.00	0.00	0.00	0.00	1600
SFT	98.75	0.69	0.00	0.56	0.00	0.00	1600
SFT + GRPO	98.31	1.06	0.00	0.62	0.00	0.00	1600

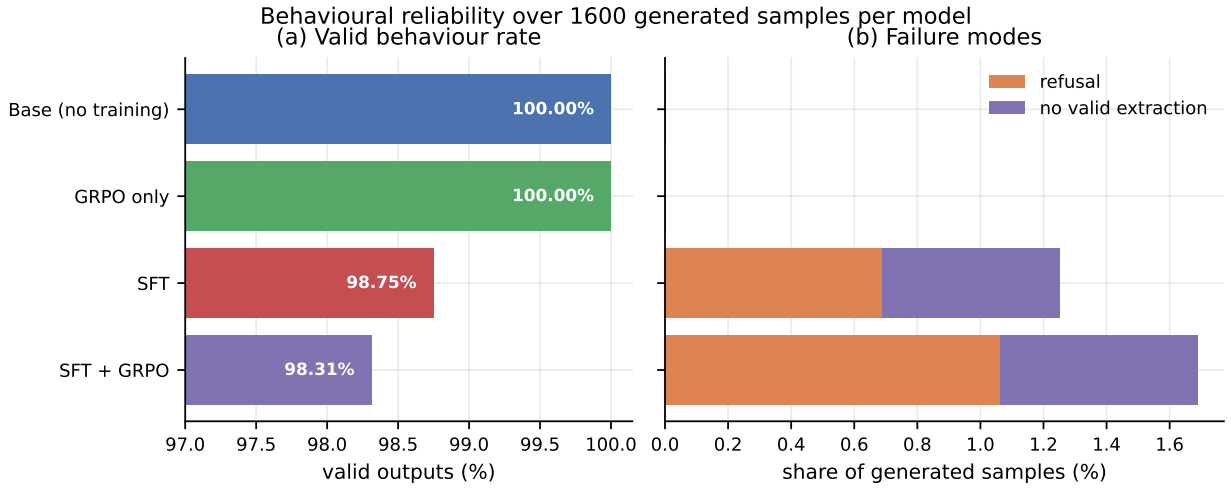

Figure 3: (a) Valid-output rate, on a truncated axis to make the 1.25–1.69 point difference visible. (b) Composition of the remaining mass by failure category.

applies directly: the channel cancels in Equation (2) and contributes nothing to the update. The same inference is not available for the untrained policy, whose format component averages -0.1525 with standard deviation 0.0050 and is therefore not at its floor; that channel remains informative. Consistently, the within-group standard deviation of the total reward is 0.124 for the fine-tuned policy against 0.204 for the untrained one, and the policy moves less (KL 0.0030 over 58 steps, against 0.0017 over 56 steps; both small in absolute terms).

One caveat remains. Our logging interval was 20 steps, so each run contributes two intermediate records plus a final summary; the statistics in Table 4 therefore describe each run's average behaviour rather than its trajectory, and we cannot say at which step saturation began. Because the floor argument above holds for each logged batch independently, this limits the temporal resolution of our observation but not its validity. Logging within-group variance per channel directly — the instrumentation we recommend in Section 7 — would remove the limitation entirely and is the one change we would make to our own pipeline.

7 Discussion

What the results support, and what they do not. Our measurements support three statements. Accuracy did not improve at any stage of this pipeline on this evaluation. One reward channel was constant across samples during the SFT + GRPO run and therefore inactive. And supervised fine-tuning introduced a failure mode, completions that announce an answer they never emit, that the untrained model does not exhibit. They do not support the stronger claim that reward-channel decomposition is inferior to multi-agent verification in general: we compare against the untrained model and against GRPO without fine-tuning, not against a system with a separately trained verifier. Nor do they establish truncation as the cause of the accuracy loss. Truncation is consistent with every diagnostic we observe, but the experiment that would establish it — rerunning the pipeline with a larger generation budget — is the one we did not

Table 4: GRPO training diagnostics, averaged over logged steps. For the fine-tuned policy the format component of the monitor channel is constant at its floor ($\mathrm{sd} = 0.0000$) and completions sit at the 512-token limit; for the untrained policy both quantities vary and completions stay well below the limit.

Run	mean reward	reward sd	format reward	completion length	KL
SFT + GRPO	-0.0672	0.1242	-0.2000	511.6	0.0030
GRPO only	-0.0619	0.2042	-0.1525	403.5	0.0017

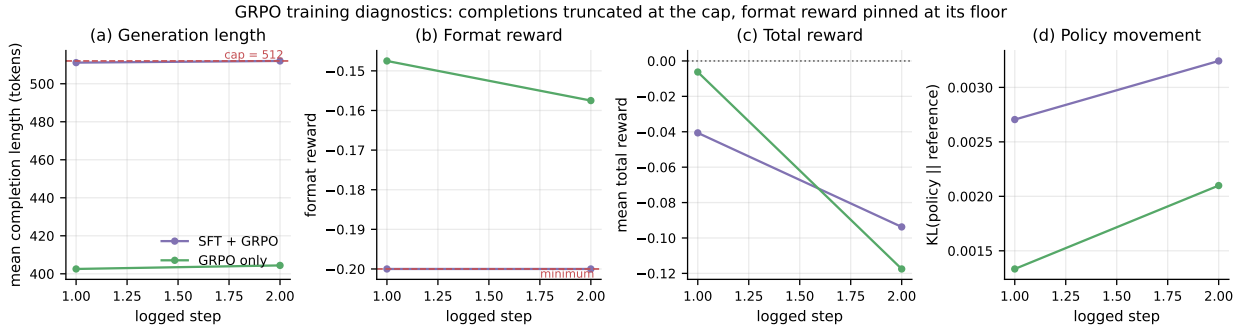

Figure 4: GRPO diagnostics for both runs. (a) Mean completion length against the 512-token limit (dashed). (b) The format component of the monitor channel against its floor (dashed). (c) Total reward. (d) KL from the reference policy. The fine-tuned policy sits at both bounds throughout; the untrained policy does not.

have time to run (Section 8).

The cost of collapsing roles. The saving from expressing roles as reward channels is real: no verifier is trained or served, and the channels cost one function call per completion. The cost is observability. In a system of separate agents, a verifier that has stopped discriminating is visible in its outputs, and a practitioner would notice. Expressed as a reward channel, the same condition produces a constant offset in the aggregate reward, which reads as a harsh but functioning penalty. The training curve in our SFT + GRPO run does not look pathological; the total reward has variance, the loss is small, the KL is finite. Only the per-channel statistics reveal that a third of the reward design was inert.

A recommended diagnostic. The instrumentation this suggests is inexpensive. For each channel k and each group, $\log \mathrm{sd}(r_1^{(k)}, \ldots, r_G^{(k)})$ alongside the aggregate, and report the fraction of groups for which it is zero. A channel that is zero-variance in most groups is contributing nothing regardless of its mean, and should be treated as a bug in the reward design rather than as a signal. This is the decomposed-reward analogue of the all-correct/all-incorrect group filtering that Yu et al. (2025) apply to the aggregate reward.

Interaction between generation budget and supervision length. Our SFT data consists of long chain-of-thought traces, while training and evaluation used a 512-token budget. The resulting model learned to produce reasoning that does not terminate within the budget it is given, which harms it under exactly the conditions it is scored in. This suggests treating the generation budget as a property of the data rather than an inference detail: if the supervision traces are long, the budget used in training, in reward computation and at evaluation must accommodate them, or the reward channels that depend on answer extraction will saturate.

8 Limitations

Single base model and single dataset. All results use DeepSeek-Math-7B-Instruct on one experiment split of OpenMathReasoning. Effects of RL on mathematical reasoning are known to vary substantially across model families (Yue et al., 2025), so we do not generalise beyond this setting.

The RL configuration is minimal. With $G = 2$ and standard-deviation normalisation, every non-degenerate group yields advantages of exactly ± 1, so our GRPO runs exercise a degenerate case of the algorithm; group sizes of 8–64 are standard. Combined with 56–58 optimiser steps, this means our experiments cannot support conclusions about GRPO's behaviour in typical configurations. What they do support is the algebraic claim of Section 4.2, which holds for any G, and the observation that the condition occurred without appearing in any aggregate-level statistic.

Single run per condition. Each condition was trained and evaluated once. We report sampling uncertainty over problems but not variation over training seeds, so we cannot separate seed variance from the effects we describe.

No paired significance tests. Our evaluation code retained aggregate rates rather than per-problem outcomes, so we report Wilson intervals but cannot run McNemar tests between conditions on the same problems. The base versus SFT + GRPO comparison is supported by disjoint intervals; intermediate comparisons are not resolved.

Coarse training logs. The GRPO logging interval yielded two intermediate records per run. We therefore characterise average training behaviour rather than its trajectory, and we did not log per-channel within-group variance directly.

The causal claim is untested. We did not rerun SFT or GRPO with a larger generation budget. Until that experiment is run, truncation is the most consistent explanation of our diagnostics rather than a demonstrated cause.

Small evaluation set. With $n = 200$, differences below roughly five percentage points are not resolvable, and we report a single benchmark rather than an external one such as MATH-500 or AIME.

9 Conclusion

We studied a construction that expresses solver, verifier and monitor roles as separate reward channels over a single policy's own generations, avoiding the inference cost of a multi-agent system. On olympiad-level problems with DeepSeek-Math-7B-Instruct, the pipeline did not improve accuracy: pass@1 decreased from 8.5% for the untrained model to 4.5% after supervised fine-tuning and 2.0% after the subsequent GRPO stage, while GRPO alone left the base model unchanged within measurement error. Training diagnostics identify a mechanism specific to the decomposition. When the generation budget truncates completions before an answer is emitted, the monitor channel takes the same value for every sample and cancels in the group-relative advantage, contributing nothing to learning while appearing in the logs as a constant penalty. We report this alongside behavioural metrics that localise the resulting degradation, and we recommend logging per-channel within-group variance as a routine diagnostic for group-relative methods with decomposed rewards.

Ethics statement

This work uses only publicly available mathematical datasets, without personal or sensitive information, and involves no human subjects or animals. We fine-tune and evaluate a publicly released language model for research purposes only, with no deployment. The main risk we see is misinterpretation: we report a negative result obtained with one model, one dataset and a single training seed, and it should not be read as evidence that reinforcement learning or verifier-style rewards are ineffective in general. We state the scope of each claim in Sections 7 and 8.

Reproducibility statement

To facilitate reproducibility, Section 5 gives the base model, data filtering and split procedure, and all training and decoding settings, and Section 4 defines each reward channel with its exact values (Table 1). Training and evaluation use separate extraction and behaviour-classification implementations, as specified in Sections 4.1 and 5. Three details affect exact reproduction: supervised fine-tuning was interrupted by a wall-clock limit and resumed from the step-4000 checkpoint; the GRPO log contains two intermediate records and a final training summary per run, summarized in Table 4; and our evaluation code retains aggregate rates rather than per-problem outcomes, which is why we report interval estimates but no paired tests. The supplementary materials contain training and evaluation code, configuration files, evaluation reports and available training logs. Completion-level and per-group reward records were not retained.

Use of large language models

Generative AI tools assisted with manuscript drafting. The authors checked the AI-assisted material and take responsibility for the final text, claims and figures. This assistance did not generate new experimental data or training runs.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. 2021.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning (ICML)*, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary LLMs. In *International Conference on Learning Representations (ICLR)*, 2024.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations (ICLR)*, 2022.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in LLM reinforcement learning via entropy-guided advantage shaping. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=kiXFIESZKv.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*, 2025.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research (TMLR)*, 2023-August, 2023. ISSN 2835-8856.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Chenxi Liu, Junjie Liang, Yuqi Jia, Bochuan Cao, Yang Bai, Heng Huang, and Xun Chen. Explore data left behind in reinforcement learning for reasoning language models. *arXiv preprint arXiv:2511.04800*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with OpenMathReasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Andries Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? a look at multi-agent debate strategies for LLMs. In *International Conference on Machine Learning (ICML)*, 2024. URL https://proceedings.mlr.press/v235/smit24a.html.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://aclanthology.org/2024.acl-long.510/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023. URL https://openreview.net/forum?id=s4xIeYimGQ.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=Ccwp4tFEtE.

A Supervised fine-tuning converged normally

Figure 5 shows the SFT training and validation loss. Training loss fell from 1.0911 to 0.6303 and validation loss decreased monotonically at every evaluation point (0.6922, 0.6623, 0.6482, 0.6402, 0.6344, 0.6308, 0.6282, 0.6266 at steps 500 through 4,000). The run therefore optimised its objective as intended; the degradation reported in Section 6.1 is not an optimisation failure but a consequence of what that objective rewards under a 512-token budget.

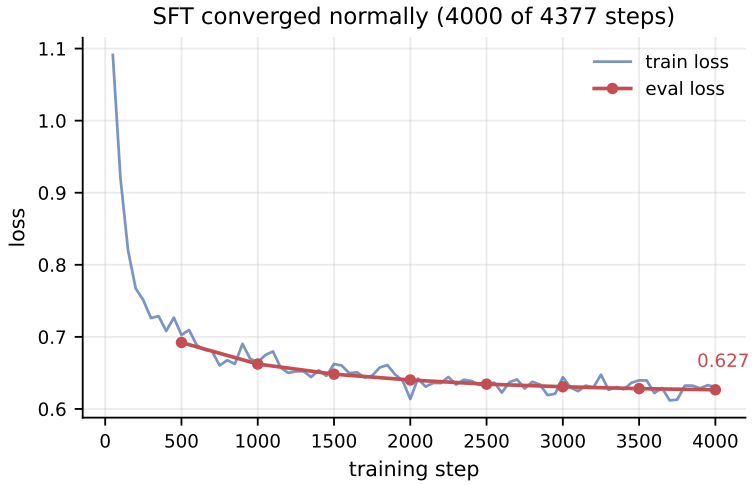

Figure 5: SFT training and validation loss. Validation loss decreases monotonically throughout.

B Per-channel reward decomposition

Figure 6 shows the mean and standard deviation of each reward channel across logged steps for both GRPO runs. The format component of the monitor channel is the only one with zero variation, and only for the fine-tuned policy.

C Effect of evaluation sample size

An earlier round of evaluation used 50 problems per condition. Figure 7 compares those estimates with the $n = 200$ estimates reported in the main text. At $n = 50$ the intervals are wide enough that the conditions are indistinguishable, and two of the point estimates move by more than two percentage points when the sample is enlarged. We report only the $n = 200$ results in the main text.

D Complete training diagnostics

Table 5 reports every quantity logged during the two GRPO runs, averaged over logged steps with the standard deviation across those steps. The rows to read together are `reward_format` and `completion_length`: for the fine-tuned policy the format channel is constant at its floor (-0.2000, $\mathrm{sd} = 0.0000$) while completions sit at the 512-token limit

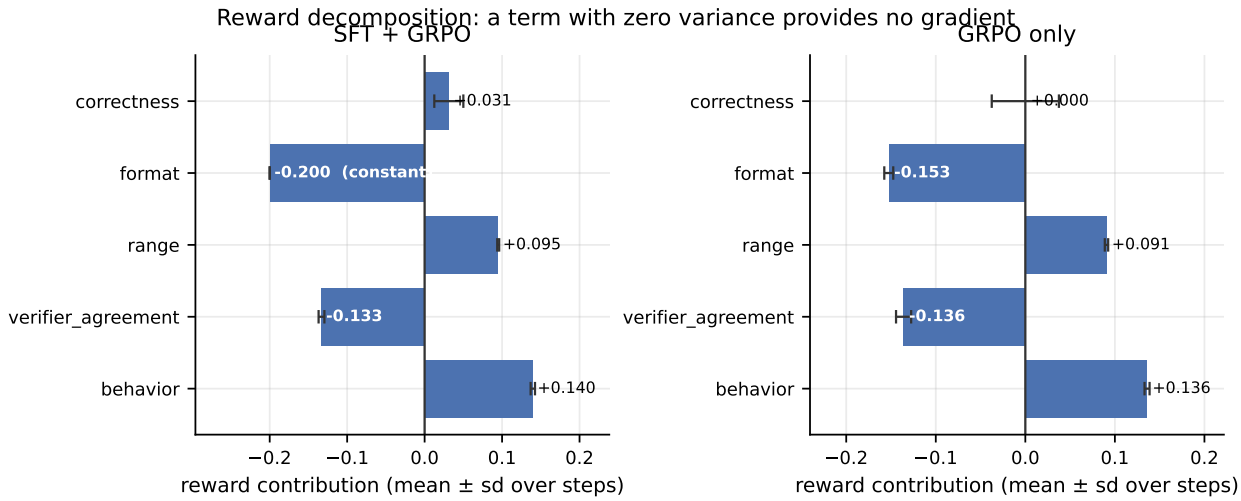

Figure 6: Reward channel contributions for both GRPO runs (mean $\pm$ sd over logged steps). A channel with zero variance cannot contribute to a group-relative update.

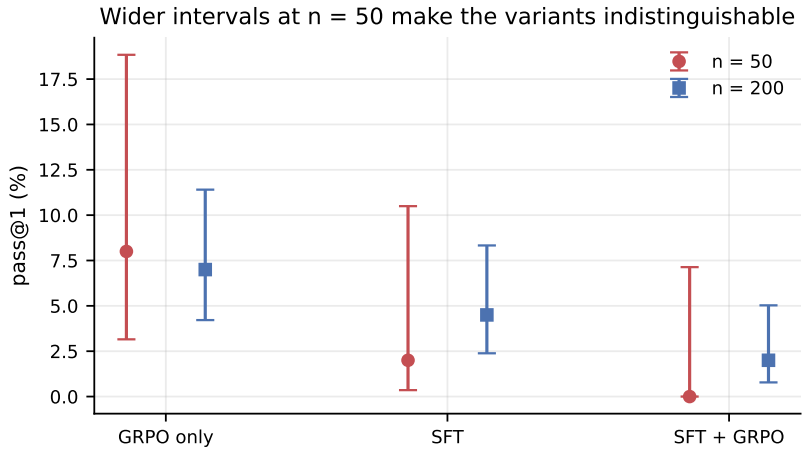

Figure 7: pass@1 with 95% Wilson intervals at $n = 50$ and $n = 200$ for the three trained conditions.

on every logged step, whereas for the untrained policy both quantities vary and completions average 403.5 tokens. The correctness channel is near zero for both runs, consistent with few extractable answers under this generation budget.

E Complete hyperparameter listing

Table 6 lists every setting used in the reported runs. The two values in bold are the ones our analysis turns on.

Note that the data filter admits target solutions of up to 1,024 tokens while generation is capped at 512, so the model is supervised on sequences up to twice the length it is permitted to produce.

Table 5: Complete GRPO diagnostics, mean ± standard deviation over logged steps (20-step logging interval; 58 steps for SFT + GRPO and 56 for GRPO only). A channel with $\mathrm{sd} = 0.0000$ takes the same value for every sample and therefore cancels in the group-relative advantage.

Quantity	SFT + GRPO	GRPO only
Total reward	-0.0672 ± 0.0266	-0.0619 ± 0.0556
Within-group reward sd	0.1242 ± 0.0013	0.2042 ± 0.0115
r_{corr} (solver)	0.0312 ± 0.0188	0.0000 ± 0.0375
r_{fmt} (monitor, format)	-0.2000 ± 0.0000	-0.1525 ± 0.0050
r_{ver} (verifier)	-0.1331 ± 0.0038	-0.1359 ± 0.0084
r_{beh} (monitor, behaviour)	0.1397 ± 0.0028	0.1359 ± 0.0028
Completion length (tokens)	511.56 ± 0.44	403.48 ± 0.93
Logged steps at the 512-token cap	100%	0%
KL from reference	0.0030 ± 0.0003	0.0017 ± 0.0004
Optimiser steps	58	56
Wall-clock time	2.58 h	2.25 h

Table 6: Hyperparameters for the reported DeepSeek-Math pipeline.

Setting	SFT	GRPO
Base model	DeepSeek-Math-7B-Instruct	
Adapter	LoRA $r = 16$, $\alpha = 32$	LoRA $r = 16$, $\alpha = 32$
LoRA target modules	`q,k,v,o,gate,up,down_proj`	
Learning rate	1.80×10^{-5}	5.0×10^{-6}
Optimiser	AdamW	AdamW
Precision	bf16	bf16
Gradient checkpointing	yes	yes
Per-device batch size	—	2
Gradient accumulation	—	4
Epochs	1	≈ 1 over mined prompts
Optimiser steps	4,377 (planned)	58 / 56
KL coefficient β	—	0.04
Group size G	—	**2**
Max prompt length	—	384
Max completion length $L_{\max}$	**512**	**512**
Evaluation interval	500 steps	—
Logging interval	—	20 steps
Data (exp2 split)		
Problems sampled N	50,000 (seed 42)	
Reasoning-quality rating	$[1, 5]$	
Max input / output length (filter)	4,096 / 1,024 tokens	
Split (train/val/mining/test)	35,021 / 5,018 / 4,949 / 5,012	
Mined prompts per GRPO run	≤ 250	
Evaluation		
Test problems	200	
maj@N samples	8	
Temperature / top-p	0.7 / 0.95	
Max new tokens	512	
Hardware	1× A100 80 GB	
CPU / host memory	8 cores / 96 GB	16 cores / 192 GB